Hardware-Aware Quantum Optimization

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

von
Thore Thassilo Gerlach
aus
Kreuztal

Bonn, 2025

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

Gutachter/Betreuer:	Prof. Dr. Christian Bauckhage
Gutachter:	Prof. Dr. Stefan Wrobel
Tag der Promotion:	11.09.2025
Erscheinungsjahr:	2025

Abstract

Quantum Optimization (QO) is emerging as a promising approach to tackle hard combinatorial optimization problems by leveraging the unique computational properties of quantum hardware. In recent years, QO has gained traction in Machine Learning (ML), where problems such as clustering and vector quantization can be naturally expressed as Quadratic Unconstrained Binary Optimization (QUBO) problems. These ML applications often involve complex, high-dimensional data, which directly influences both the structure and scale of the resulting QUBO formulations. However, current quantum devices fall under the category of Noisy Intermediate-Scale Quantum (NISQ) hardware, which imposes significant limitations on qubit counts and small error rates. This dissertation investigates the impact of data complexity and data scale—as encountered in ML settings—on the effectiveness and feasibility of QO on NISQ devices. We propose novel, hardware-aware methods to overcome these challenges.

The first part of this thesis focuses on the effect of data complexity for QO. We analyze how problem structure—particularly when QUBO formulations are derived from ML tasks—affects quantum solvability. We identify the spectral gap as a key factor in the success of adiabatic quantum algorithms and provide a theoretical and empirical analysis linking data-induced complexity to optimization performance.To address noise sensitivity and limited precision, we introduce a preprocessing framework based on a principled Branch-and-Bound algorithm that reduces dynamic range while preserving global optima. This improves robustness on both quantum annealers and classical solvers.

In the second part, the thesis addresses challenges introduced by data scale. First, we propose a recursive Divide-and-Conquer strategy that breaks down large QUBO problems into tractable subproblems, demonstrated through an application to Bundle Adjustment in 3D scene reconstruction. Second, we introduce a variable and constraint generation method inspired by column generation for Integer Linear Programming, enabling dynamic QUBO size growth. This technique is successfully applied to Multi-Agent Pathfinding, where it scales well on NISQ devices while maintaining compatibility with classical solvers. Finally, we explore QUBO reformulations that reduce problem size through compact encodings and cyclic expansion, enabling the use of quantum hardware for tasks such as FPGA placement in chip design.

Across all contributions, the developed algorithms are evaluated both theoretically and empirically, with experiments conducted on real quantum and quantum-inspired hardware. These methods significantly improve solution quality, scalability, and hardware compatibility, paving the way for more effective use of QO in practical settings. While tailored for NISQ era constraints, the results remain relevant for fault-tolerant quantum computing, offering insights into the long-term viability of QO across hardware generations.

Acknowledgements

I am deeply grateful to the many people—mentors, colleagues, friends, and family—whose support, encouragement, and understanding made this thesis possible.

First and foremost, I would like to thank my supervisors, Prof. Dr. Christian Bauckhage and Prof. Dr. Stefan Wrobel, for accepting me as their PhD student. I would especially like to extend my thanks to Christian for the academic freedom he afforded me, and his trust in my ability to conduct independent research.

Secondly, I would also like to thank my colleagues Dr. Nico Piatkowski and Sascha Mücke who were always open to engaging discussions. I am especially grateful to Nico for his mentorship, which significantly contributed to my professional growth. Also, a big thanks to my colleagues at Fraunhofer IAIS and the University of Bonn for making office days and work trips not just productive, but genuinely fun—both in and outside of research.

Finally, I would like to thank my family and friends for their unwavering support, encouragement, and presence throughout this journey. I am especially grateful to my parents and siblings for their love and belief in me, and to my friends for bringing balance and perspective to life beyond research.

Contents

CHAPTER 1

Introduction

Optimization plays a fundamental role in many application areas, such as scientific and engineering disciplines, enabling efficient decision-making in complex systems. Whether in logistics [1], finance [2], healthcare [3], energy systems [4], transportation [5], or Artificial Intelligence (AI) [6], optimization techniques help in finding the best possible solutions under given constraints. One can generally use two different categorizations: in continuous optimization, the decision variables can take any real value within a specified range while discrete optimization deals with problems where variables are discrete, such as integers. Methods for finding a (sub)optimal solution have been extensively studied and applied to a wide range of real-world problems [7–10]. However, as problem sizes grow and constraints become more intricate, many algorithms face increasing computational challenges, particularly in high-dimensional and discrete settings.

Machine Learning (ML) [11] has revolutionized how we process and analyze information and is a perfect example for being exposed to the aforementioned challenges. It enables systems to learn from data, recognize patterns and make predictions or decisions without explicit programming. ML can be understood as the process of fitting mathematical models to data—known as *training*—with optimization serving as the foundational backbone of this process. Training ML models typically entails minimizing a loss function, tuning hyperparameters, or solving complex combinatorial problems. Exemplary tasks include the classification of images, detecting fraud in banking transactions or natural language processing, such as translating text from one language to another. Arguably, the most prominent approach for tackling these tasks is Deep Learning (DL) [12], which aims to simulate the complex decision-making structures of the human brain with an Artificial Neural Network (ANN). Using Gradient Descent (GD) as the optimization method of choice, DL models can be efficiently trained using *backpropagation*—a method fundamentally based on linear algebra operations [13]. The availability of special-purpose hardware, capable of massively parallelizing such operations, has significantly expanded the scope and scale of DL applications in recent years. That is, large DL models have shown impressive performance in playing games [14], protein folding [15], or generating text [16], images [17] and videos [18]. These state-of-the-art DL models consist of up to billions of parameters and are trained on millions of data samples. Training such large-scale models is becoming progressively more expensive and inefficient, demanding increasing amounts of time and computational resources to navigate the exponentially expanding parameter space.

As datasets continue to grow in size and complexity, optimizing ML workflows becomes computationally demanding, requiring innovative techniques to efficiently process large-scale data. With *data*

scale, we refer not only to the dataset size, but also to the underlying dimensionality of single data points, while *data complexity* solely refers to intrinsic properties not concerned with size, such as noise or the presence of outliers. We use the term data complexity, since the complexity/hardness of an ML task is largely dependent on characteristics of the underlying data. In general, data serves as the foundation of both optimization and ML, influencing solution quality and computational feasibility. The scale and complexity of data impact how well an optimization algorithm performs and whether an ML model generalizes effectively—that is, how accurately it captures the true underlying data distribution. Large scale and high complexity introduce challenges related to sparsity, feature selection, and noise, making data preprocessing and transformation crucial steps in modern computational workflows. Thus, understanding the relationship between data properties and optimization techniques is essential for designing robust and efficient algorithms.

Quantum Computing (QC) [19] offers a new paradigm for solving computationally challenging problems, leveraging the principles of quantum mechanics to process information in fundamentally different ways than classical computers do. It offers the potential for solving certain computational tasks in a reasonable time, where classical computers would need decades or even centuries to come up with a solution. Unlike the usage of classical bits for representing information in classical computers, quantum systems operate using quantum bits (*qubits*). Unlike classical bits, which exist in a definite state (0 or 1), qubits can exist in a superposition of states—a concept that defies human intuition. Measuring the state of a qubit also leads to the observation of a definite state, but only with a certain probability. More specifically, measuring a quantum system consisting of multiple qubits can be seen as a discrete probability distribution over an exponentially large space. Another unusual property without an analogue in our macroscopic world is quantum entanglement. Denoted by Albert Einstein as a "spooky action at a distance" [20], it describes the interaction between different qubits, such that the state of one qubit instantly influences the state of the other. Due to these two quantum phenomena, highly entangled qubit systems are intractable to be simulated with a classical computer, while quantum hardware can efficiently manipulate such quantum states. The key question in QC is how to leverage the implicit manipulation of a quantum state to achieve meaningful results.

The perhaps most well-known example for a quantum algorithm with a provably asymptotic speed-up is the *quantum fourier transform* [21], which offers an exponential run-time advantage over the classical counterpart. This insight has been used by Peter Shor, who developed a polynomial-time QC algorithm for prime factorization [22], which is assumed to be unsolvable in a reasonable time with a classical computer. Many cryptographic standards are based on this assumption, leading to QC posing a potential threat—at least in theory. Another famous example is unordered data base search [23], which has a provable quadratic speed-up over classical methods. These two algorithms solve a classical task by outsourcing computationally intensive problems to quantum computations. However, quantum computers are also suited for working with data that is inherently quantum in nature. Especially in obtaining accurate simulations of quantum systems, inter alia used for drug discovery [24] and material science [25], QC can be very beneficial. Another application field is Quantum Machine Learning (QML) [26], which explores how quantum algorithms can enhance ML tasks by encoding and processing data in quantum states. The underlying data can be either quantum or classical. However, efficient classical-to-quantum data conversion is often the bottleneck when applying QML algorithms.

One of the most promising application areas using classical data is Quantum Optimization (QO) [27], which addresses problems where classical techniques struggle due to exponential complexity. Many optimization problems, particularly those in Combinatorial Optimization (CO), can be mapped to Quadratic Unconstrained Binary Optimization (QUBO) [9] problems and the equivalent *Ising model* [28].

These formulations can be conveniently encoded into a *quantum Hamiltonian*, which describes the total energy of a quantum system and how it evolves over time. Through quantum mechanical principles, an optimal configuration of the optimization problem at hand is obtained by searching for a quantum state with minimal energy regarding this Hamiltonian. Dependent on the underlying problem setting, this search can be conducted efficiently in the quantum realm, e. g., by exploiting quantum tunneling effects.

Although the aforementioned advantages of QC over classical digital computing for certain application areas may sound very appealing, these insights are of a theoretical nature, and a real-world advantage on quantum hardware has not (yet) been proven. This is due to the limitations of current and near-term quantum devices, often referred to as Noisy Intermediate-Scale Quantum (NISQ) hardware [29]. These devices are prone to noice-induced errors and offer limited qubit counts, preventing the solvability of large-scale problems. NISQ hardware does not yet possess full error correction capabilities, making it essential to develop algorithms that are resilient to noise. Understanding how to leverage NISQ devices effectively thus requires strategies to mitigate noise and to outsource computationally hard tasks with a problem size suited for the limited number of qubits.

Due to the close connection between problem structure/dimensionality and underlying data, this thesis investigates the interplay between data and the performance of QO on NISQ devices. Specifically, we first ask the question of how the underlying data complexity affects the solvability of QUBO problems with QO algorithms. Considering the inherent noise in NISQ hardware, we analyze the impact of different data properties on computational efficiency and solution quality and propose generally applicable mitigation techniques to counteract possible negative effects. Additionally, the limited qubit availability and intermediate scale of current quantum devices impose constraints on problem sizes and encoding strategies. We explore how these constraints influence the effectiveness of QO and adopt theoretically sound methods to improve scalability while preserving the integrity of the optimization process. Our developed methods are adapted to real-world large-scale use-cases, where high dimensionality arises from the underlying data, and are evaluated on NISQ hardware in order to demonstrate their effectiveness.

Even though we concentrate on NISQ devices and their limitations, our insights can be directly transferred to post-NISQ quantum computers. The largest quantum hardware vendors (optimistically) promise systems with up to thousands of (nearly) error-corrected qubits until 2033[1,2,3]. This number may sound impressive and can definitely be advantageous for certain application areas, such as cryptography and quantum simulations, but solving large-scale optimization problems is still way out of reach. Further, with current hardware approaches, noise effects can still corrupt complex computations. Thus, our proposed methods can also be of great benefit for upcoming quantum hardware in the long term.

1.1 Thesis Outline

This thesis investigates the impact of data on QO performance in the NISQ era, and is split into two parts preceded by an extensive background. In the first part (Chapters 3 and 4), we investigate the effect of data complexity for QO, respecting the noisiness of NISQ devices. The second part (Chapters 5 to 7) bridges the gap to the limited scale, which examines the impact of data scale for QO performance. A structural high-level overview of this thesis can be found in Figure 1.1, while the introductions of

[1] https://www.meetiqm.com/technology/roadmap (last accessed September 19, 2025)
[2] https://www.ibm.com/roadmaps/quantum (last accessed September 19, 2025)
[3] https://quantumai.google/roadmap (last accessed September 19, 2025)

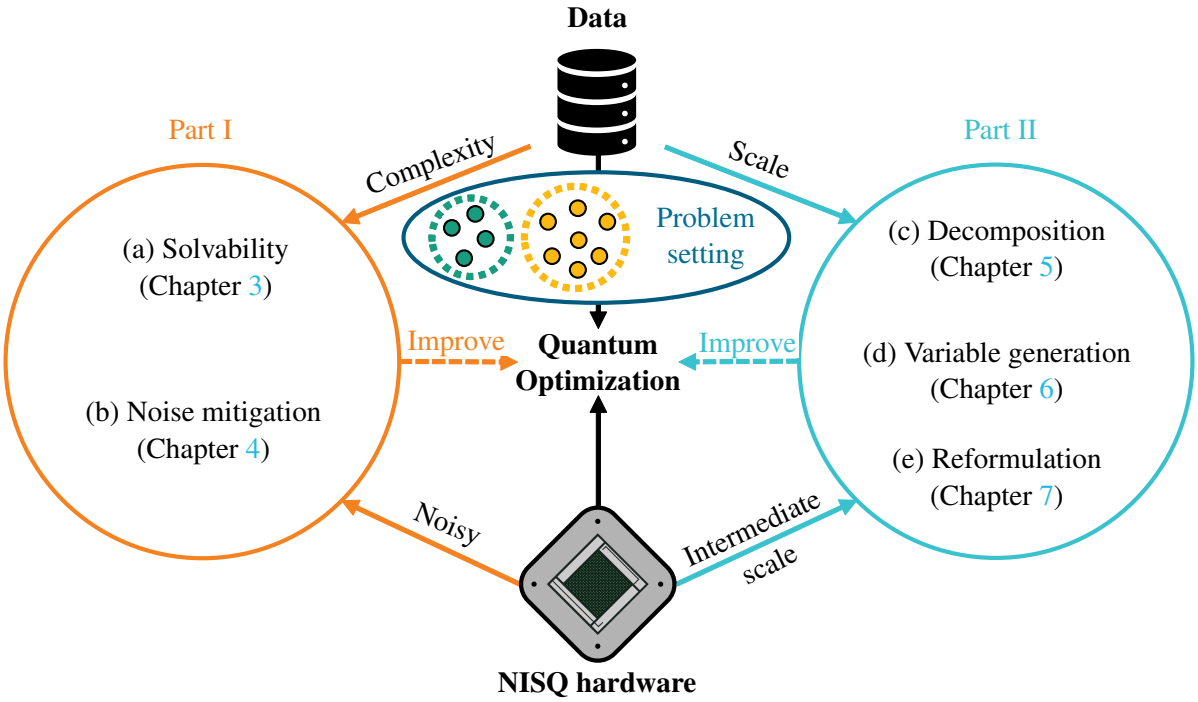


Figure 1.1: Structural overview of this thesis. We investigate the relation between the underlying data of a problem setting and the consequential performance of QO techniques for NISQ devices. A schematic representation of clustering is used for representing a generally hard optimization task. In the first part of this thesis, we delve into the effect of data complexity on the solvability for QO (a) and develop a generally applicable noise-reduction method, coping with the proneness to errors of current NISQ devices (b). Furthermore, we propose theoretically sound problem size reduction techniques in part two, which are indispensable due to the possibly huge scale of data for real-world use-cases and the limited size of NISQ hardware. An efficient top-down problem decomposition algorithm is developed (c), while we also consider a bottom-up variable generation approach for controlling the problem size (d). Finally, we explore size reduction by using efficient reformulations of the problem (e). Overview figures for the single chapters—similar to this one—can be found in the respective introductions. Our developed methods help to improve the performance of QO in the NISQ era and beyond.

the single chapters contain more low-level figures for describing the methods developed therein. The following paragraphs provide a more detailed description of the various chapters.

Before we get to the first part, we start off with establishing foundational concepts necessary for the subsequent chapters in Chapter 2. It begins with an overview of essential mathematical notation, focusing on linear algebra, binary and complex vector spaces. The discussion transitions into optimization with an emphasis on QUBO and its role in combinatorial problem-solving. Next, ML is introduced, detailing the three primary paradigms—Supervised Learning (SL), Unsupervised Learning (UL), and Reinforcement Learning (RL)—alongside applications such as Support Vector Machines (SVMs) [30], clustering [31], and Vector Quantization (VQ) [32]. The chapter then shifts to QC, presenting its fundamental principles, including qubits, superposition, entanglement, and quantum measurement. The two primary models of QC—Quantum Gate Computing (QGC) and Adiabatic Quantum Computing (AQC) [33]—are explored, along with their applications for QO. Finally, we discuss the availability, advantages and limitations of current quantum hardware, particularly in the NISQ era, setting the stage for the thesis's investigation

into practical QO strategies.

Part I We begin the first part by investigating how certain data characteristics affect the efficiency and feasibility of solving QUBO problems on quantum devices in Chapter 3. A central focus is the Spectral Gap (SG), a critical parameter influencing the success of AQC. We examine how various ML problem structures—such as clustering and SVMs—interact with QUBO formulations and influence the solution quality. The findings provide insights into the fundamental relationship between problem structure and quantum solvability, guiding the development of more efficient QO methods.

Beyond solvability concerns, NISQ devices are prone to computational errors that can significantly distort optimization outcomes. In Chapter 4, we thus explore the challenges introduced by noise in quantum hardware for QO and propose a framework for assessing and mitigating these effects. A key focus lies on parameter precision in QUBO instances, which directly impacts the robustness of quantum computations. For quantifying the precision needed for faithfully representing the problem parameters, we use the Dynamic Range (DR) often used in signal processing. We introduce preprocessing techniques that optimize QUBO formulations for hardware resilience, reducing the risk of false optima caused by numerical errors. These techniques are based on using a Markov Decision Process (MDP) formulation and the development of a principled Branch-and-Bound (B&B) algorithm to navigate through the exponentially large search space, preserving optimal solutions of the underlying problem. This leads to a generally applicable QUBO preprocessing procedure for mitigating hardware noise, in opposition to existing methods specifically tailored towards certain problem structures. Through experimental evaluations on NISQ devices and Field-Programmable Gate Array (FPGA)-based quantum emulation hardware, the chapter demonstrates the effectiveness of these noise-mitigation strategies in improving optimization reliability and reduction of resources for classical hardware solvers.

Part II Due to hardware constraints, many optimization problems exceed the feasible size for NISQ devices, necessitating strategies to reduce the problem size without compromising solution quality. Chapter 5 introduces an iterative top-down Divide-and-Conquer (D&C) approach that decomposes large QUBO instances into smaller, manageable subproblems. Unlike traditional methods that process problems holistically, this technique progressively refines solutions by solving smaller subproblems and reintegrating their results. Special emphasis is placed on underlying constraint structures, leading to efficient recombination strategies. The method is particularly beneficial for ML-related CO tasks, where data scale often leads to prohibitively large QUBO matrices. A key application of this method is in computer vision, specifically in 3D scene reconstruction using Bundle Adjustment (BA) [34], where multiple camera images are analyzed to extract 3D geometric structures. By reinterpreting the keypoint extraction step of BA as a VQ problem, the D&C approach enables efficient quantum processing while preserving global correlations between image features. The method is evaluated on satellite image data, which have a very high pixel resolution, leading to huge image feature dimensionality. Evaluating our approach on current quantum hardware, we demonstrate how quantum techniques can improve large-scale optimization tasks in vision-based applications while remaining compatible with current NISQ hardware constraints.

In contrast to the top-down method followed in Chapter 5, Chapter 6 builds on bottom-up decomposition strategies. We explore an alternative approach to controlling QUBO problem size through dynamic variable and constraint generation. Instead of solving the entire problem upfront, the method begins with a small subset of variables, gradually expanding the formulation by introducing new variables and

constraints as needed. Inspired by *column generation* [35] techniques for Integer Linear Programming (ILP), the approach leverages Lagrangian Relaxation (LR) [36] to compute efficient lower bounds, while QO is used to determine upper bounds. Using these bounds, we prove an optimality criterion that tells us when our generated variables already obtain an optimal solution to the underlying problem, leading to a potentially drastic problem-size reduction. A key application of this method is in Multi-Agent Pathfinding (MAPF) [37], where it is used to optimize collision-free routes for multiple agents in complex environments. By iteratively refining the optimization process, this approach enables scalability while preserving solution quality on quantum hardware. We show the compatibility with state-of-the-art methods by introducing hardware-aware QUBO formulations.

The second part finishes with Chapter 7 where alternative QUBO encodings are explored to enhance efficiency and scalability. Instead of iteratively solving subproblems or expanding the variable set, the approach focuses on reformulating the problem structure to achieve more compact and sparse representations. For CO tasks such as the Quadratic Assignment Problem (QAP) [38], a logarithmic-sized encoding is introduced, reducing the number of required decision variables. Additionally, the chapter presents our developed Cyclic Expansion algorithm for unbalanced QAPs inspired by the α-expansion method [39] in classical optimization. This iterative approach refines solutions by optimizing over cyclic permutations, bypassing the need for explicit constraint incorporation. The method is applied to FPGA-placement problems in chip design [40] using NISQ devices, demonstrating its ability to handle large-scale optimization tasks while remaining adaptable to quantum hardware limitations.

Finally, this thesis is concluded in Chapter 8 with summarizing and discussing the most important results. We highlight future work and provide an outlook on how our developed methods can also improve QO performance beyond the NISQ era, particularly for fully fault-tolerant quantum computers.

1.2 Overview of Publications

This thesis is based on a number of scientific peer-reviewed papers published between 2022 and 2025, presented in chronological order.

[41] N. Piatkowski et al., “Towards bundle adjustment for satellite imaging via quantum machine learning”, *Proceedings of the 25th International Conference on Information Fusion (FUSION)*, IEEE, 2022 1, DOI: https://doi.org/10.23919/FUSION49751.2022.9841388

[42] T. Gerlach et al., “FPGA-placement via quantum annealing”, *Proceedings of the 32nd ACM/SIGDA International Symposium on Field Programmable Gate Arrays (ISFPGA)*, ACM, 2024 43, DOI: https://doi.org/10.1145/3626202.3637619

[43] T. Gerlach and S. Mücke, “Investigating the relation between problem hardness and QUBO properties”, *Proceedings of the 22nd International Symposium on Intelligent Data Analysis (IDA)*, Springer, 2024 171, DOI: https://doi.org/10.1007/978-3-031-58553-1_14

[44] T. Gerlach et al., “Quantum optimization for FPGA-placement”, *Proceedings of the 2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, IEEE, 2024 637, DOI: https://doi.org/10.1109/QCE60285.2024.00080

[45] S. Mücke, T. Gerlach and N. Piatkowski, *Optimum-preserving QUBO parameter compression*, Quantum Machine Intelligence **7**.1 (2025) 1, DOI: https://doi.org/10.1007/s42484-024-00219-3

[46] T. Gerlach et al., “Hybrid quantum-classical multi-agent pathfinding”, *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, To appear, PMLR, 2025, DOI: https://doi.org/10.48550/arXiv.2501.14568

[47] T. Gerlach and N. Piatkowski, “Dynamic range reduction via branch-and-bound”, *Proceedings of the 2025 IEEE International Conference on Quantum Computing and Engineering (QCE)*, To appear, IEEE, 2025, DOI: https://doi.org/10.48550/arXiv.2409.10863

[48] T. Gerlach, S. Mücke and C. Bauckhage, “Kernel k-Medoids as General Vector Quantization”, *Proceedings of the 2025 IEEE International Conference on Quantum Artificial Intelligence (QAI)*, To appear, IEEE, 2025, DOI: https://doi.org/10.48550/arXiv.2506.04786

CHAPTER 2

Background

In this chapter, Section 2.1 establishes notation that will be used throughout this thesis, while basic concepts forming the foundation of subsequent chapters are introduced. Specifically, we will introduce (combinatorial) optimization (Section 2.2) and ML (Section 2.3), with an emphasis on the problems and models relevant to later chapters. An overview of the two main paradigms of QC is given in Section 2.4 along with an overview of QO and limitations of current quantum hardware.

2.1 Notation and Basic Definitions

The notational conventions adhered to throughout this thesis are briefly outlined in this section. Since the theory behind QC is mainly based on *complex linear algebra*, we will concentrate on matrix/vector notation and recall basic concepts.

2.1.1 Basic Linear Algebra

Vectors are represented by lowercase boldface letters, matrices by uppercase boldface letters and their respective elements are written using their non-boldface counterparts with subscript indices. As an example, let $\boldsymbol{A}$ be an arbitrary $n \times m$ matrix and $\boldsymbol{a}$ a vector of size n, $n, m \in \mathbb{N}$. Then a_i is the i-th element of $\boldsymbol{a}$ and A_{ij} the element of $\boldsymbol{A}$ in row i and column j, where i and j are elements of the ordered set from 1 to n, i. e., $i, j \in [n] \coloneqq \{1, \dots, n\}$. The i-th row of $\boldsymbol{A}$ is denoted by $\boldsymbol{A}_i$ and the j-th column by $\boldsymbol{A}_{\cdot j}$. Note that indexing a boldface vector $\boldsymbol{a}_i$ does not denote the i-th entry of $\boldsymbol{a}$, but some enumeration of an element of a set of vectors, i. e., $\{\boldsymbol{a}_1, \dots, \boldsymbol{a}_n\} \subset \mathbb{R}^n$. More advanced indexing is obtained by defining subvectors and submatrices.

Definition 2.1 (Subindexing)**.** Let $I = \{i_1, \dots, i_k\} \subseteq [n]$, $J = \{j_1, \dots, j_l\} \subseteq [m]$, $\boldsymbol{a} \in K^n$ and $\boldsymbol{A} \in K^{n \times m}$ for some field K and $k, l, n, m \in \mathbb{N}$. Then $\boldsymbol{a}_I$ is a subvector of $\boldsymbol{a}$ defined as

$$\boldsymbol{a}_I = \left(a_{i_1}, \dots, a_{j_k}\right)^\top ,$$

and the submatrix $\boldsymbol{A}_{I,J}$ is given by

$$\boldsymbol{A}_{I,J} = \begin{pmatrix} A_{i_1,j_1} & \cdots & A_{i_1,j_l} \\ \vdots & \ddots & \vdots \\ A_{i_k,j_1} & \cdots & A_{i_k,j_l} \end{pmatrix} .$$

For an index set $I \subset [n]$, we define its complement as $I^c \coloneqq [n] \setminus I$. All vectors are assumed to be column vectors and the superscript $\top$ denotes the transpose of vectors and matrices. Addition, subtraction and equality is element-wise between vectors and matrices. We denote element-wise multiplication by $\odot$ and element-wise inequality by $\preceq$. For a quadratic matrix $\boldsymbol{A}$, we indicate its diagonal vector by $\mathrm{diag}(\boldsymbol{A})$. Further, if the argument is a vector $\boldsymbol{a}$, we define $\mathrm{diag}(\boldsymbol{a})$ to be the diagonal matrix with $\boldsymbol{a}$ as its corresponding diagonal. tr denotes the *trace* of a matrix, that is $\mathrm{tr}(\boldsymbol{A}) = \mathbf{1}^\top \mathrm{diag}(\boldsymbol{A})$ and $\mathrm{vec}(\boldsymbol{A})$ is defined as the concatenation of all rows of $\boldsymbol{A}$ into a single vector, that is $\mathrm{vec}(\boldsymbol{A}) = (\boldsymbol{A}_1^\top, \ldots, \boldsymbol{A}_n^\top)^\top$. The *euclidean norm* or the length of a real vector $\boldsymbol{a} \in \mathbb{R}^n$ is given by $\|\boldsymbol{a}\|_2 = \sqrt{\boldsymbol{a}^\top \boldsymbol{a}}$ and we simply denote it by $\|\cdot\|$ without using the subscript. We also recall the definition of the *Kronecker product*, since it is vital for QC.

Definition 2.2 (Kronecker)**.** Given two matrices $\boldsymbol{A} \in K^{n \times m}$ and $\boldsymbol{B} \in K^{n' \times m'}$ for some field K, the Kronecker product is defined as

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{pmatrix} A_{11}\boldsymbol{B} & \cdots & A_{1m}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ A_{n1}\boldsymbol{B} & \cdots & A_{nm}\boldsymbol{B} \end{pmatrix} \in K^{nn' \times mm'} .$$

Applying the Kronecker product k times is denoted by $\boldsymbol{A}^{\otimes k} = \boldsymbol{A} \otimes \cdots \otimes \boldsymbol{A}$. Note that the dimension $\boldsymbol{A}^{\otimes k}$ of grows exponentially with k.

2.1.2 Binary Vector Spaces

We denote the set of binary vectors of size n as $\mathbb{B}^n$, where $\mathbb{B} = \{0, 1\}$. The $n \times n$ identity matrix is denoted by $\boldsymbol{I}_n \in \mathbb{B}^{n \times n}$, the n-dimensional vectors consisting only of zeros and ones by $\mathbf{0}_n$ and $\mathbf{1}_n$ and the i-th standard basis vector as $\boldsymbol{e}_i^n$. The subscript/superscript is omitted if the size is clear from the context. Matrices consisting only of single binary values are denoted as outer products. That is the $m \times n$ matrix consisting only of zeros is written as $\mathbf{0}_m \mathbf{0}_n^\top$ and the matrix consisting only of ones as $\mathbf{1}_m \mathbf{1}_n^\top$. For the rest of the thesis we also need the concept of fixing certain entries in binary vectors.

Definition 2.3 (Fixed subspace)**.** Let $1 \leq m \leq n$, $I \subseteq [n]$ with $|I| = m$, $\boldsymbol{z} \in \mathbb{B}^n$, and $\boldsymbol{\zeta} \in \mathbb{B}^m$. The set of all n-dimensional bit vectors in which the variables indexed by I are fixed to the values in $\boldsymbol{\zeta}$ is denoted as

$$\mathbb{B}^n_{I \leftarrow \boldsymbol{\zeta}} \coloneqq \{\boldsymbol{z}' : \boldsymbol{z}' \in \mathbb{B}^n, \boldsymbol{z}'_I = \boldsymbol{\zeta}\} . \tag{2.1}$$

2.1.3 Complex Vector Spaces

Another important aspect for QC are complex vector spaces, so we give a quick recap about complex numbers. Every element $z \in \mathbb{C}$ can be written as $z = a + ib$, with real part $a = \mathrm{Re}(a + ib) \in \mathbb{R}$,

imaginary part $b = \mathrm{Im}(a + ib) \in \mathbb{R}$ and the imaginary unit i which fulfills $i^2 = -1$. An alternative form is given by its polar representation $z = r \cdot e^{i\varphi}$, $r \in \mathbb{R}$, $\varphi \in [0, 2\pi]$, with the complex exponential $e^{i\varphi} = \cos(\varphi) + i\sin(\varphi)$. A transformation between these different representations or coordinates is given by $a = r\cos(\varphi)$ and $b = r\sin(\varphi)$, giving rise to a two-dimensional real vector space. The complex conjugate of $z = a + ib = re^{i\varphi} \in \mathbb{C}$ is given by $z^* \coloneqq a - ib = re^{-i\varphi}$ and the absolute value by $|z| = \sqrt{z \cdot z^*} = \sqrt{a^2 + b^2} = r$.

A complex matrix $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ is called *Hermitian*—the complex extension of a symmetric matrix—if $\boldsymbol{A}^\dagger = \boldsymbol{A}$, where the superscript † indicates the combination of element-wise complex conjugation and the matrix transpose, i. e., $\boldsymbol{A}^\dagger \coloneqq (\boldsymbol{A}^*)^\top$. A Hermitian matrix $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ is said to be *positive definite* if $\boldsymbol{z}^\top \boldsymbol{A}\boldsymbol{z} > 0$ and *positive semi-definite* if $\boldsymbol{z}^\top \boldsymbol{A}\boldsymbol{z} \geq 0$ for all non-zero $\boldsymbol{z} \in \mathbb{C}^n$. The complex extension of orthogonal real matrices is given by *unitary* matrices which fulfill $\boldsymbol{A}^\dagger \boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^\dagger = \boldsymbol{I}$.

Definition 2.4 (Inner product)**.** An inner product space is a complex vector space $\mathcal{H}$ equipped with an inner product $\langle \cdot \mid \cdot \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{C}$ satisfying the following properties:

1. $\langle \boldsymbol{x} \mid \boldsymbol{y} \rangle^* = \langle \boldsymbol{y} \mid \boldsymbol{x} \rangle\,,\ \forall \boldsymbol{x} \mid \boldsymbol{y} \in \mathcal{H}$ (*conjugate symmetry*),
2. $\langle a\boldsymbol{x} + b\boldsymbol{y} \mid \boldsymbol{z} \rangle = a \langle \boldsymbol{x} \mid \boldsymbol{z} \rangle + b \langle \boldsymbol{y} \mid \boldsymbol{z} \rangle\,,\ \forall \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathcal{H},\ a, b \in \mathbb{C}$ (*linearity*),
3. $\langle \boldsymbol{x} \mid \boldsymbol{x} \rangle > 0,\ \forall \boldsymbol{x} \neq \boldsymbol{0} \in \mathcal{H}$ and $\langle \boldsymbol{0} \mid \boldsymbol{0} \rangle = 0$ (*positive definiteness*).

An inner product $\langle \cdot, \cdot \rangle$ induces a norm and hence also a distance measure given by $d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\langle \boldsymbol{x} - \boldsymbol{y} \mid \boldsymbol{x} - \boldsymbol{y} \rangle}$. A *Hilbert* space is a complete inner product space, i. e., every *Cauchy* sequence is converging w.r.t. the induced distance measure. For QC, we will consider the complex vector space $\mathbb{C}^n$, equipped with the inner product $\langle \boldsymbol{x} \mid \boldsymbol{y} \rangle \coloneqq \boldsymbol{x}^\dagger \boldsymbol{y}$ which is a Hilbert space.

2.2 Optimization

Optimization lies at the heart of many problems in computer science—particularly in ML. It involves finding the best solution from a set of feasible solutions according to a specified criterion. These problems are typically defined in terms of

- **Objective Funtion** $f : \mathbb{X} \to \mathbb{R}$: Criterion to be maximized or minimized,
- **Domain** $\mathbb{X}$: The variables that can be adjusted for optimization,
- **Constraints** $g_i : \mathbb{X} \to \mathbb{R}$: Conditions $g_i(x) \leq 0$ that define the feasible region.

Mathematically, an optimization objective can be written as

$$\min_{x \in \mathbb{X}} f(x) \tag{2.2a}$$

$$\text{s.t. } g_i(x) \leq 0,\ \forall i \in [m]\,, \tag{2.2b}$$

where m is the number of constraints and without loss of generality, we assume we want to minimize our objective function. Note that the optimum is an element of $\mathbb{R}$—we denote the corresponding optimal $x^* \in \mathbb{X}$ as an optimizer. Finding an exact (global) solution to Equation (2.2) is often intractable in practice. This can be due to the nature of the objective function, the domain and/or the constraints. Thus,

one is often satisfied with a finding an optimum in a local neighborhood $\mathcal{N}(y) \subseteq \mathbb{X}$ for a given element $y \in \mathbb{X}$. The arguably most well-known method for finding a local optimum for an arbitrary differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ involves its gradient. It is defined by

$$\nabla f(\boldsymbol{r}) = \left(\frac{\partial f}{\partial r_1}(\boldsymbol{r}), \ldots, \frac{\partial f}{\partial r_n}(\boldsymbol{r}) \right)^\top ,$$

which consists of the vector of all partial derivatives w.r.t. to every coordinate of the input. Descriptively, it points towards the steepest local ascent of f. This motivates a method for finding a local optimum: we can start from some initial position $\boldsymbol{r}^0$, compute the gradient $\nabla f(\boldsymbol{r}^0)$ and take a step into the opposite direction of the steepest ascent

$$\boldsymbol{r}^{t+1} = \boldsymbol{r}^t - \eta \nabla f(\boldsymbol{r}^t) ,$$

where $\eta > 0$ characterizes the step size (learning rate in ML) and $t \in \mathbb{N}$ indicates the current iteration. Carefully choosing η, the sequence $f(\boldsymbol{r}^t)$ converges towards a local minimum of f. This iterative method is known as GD and a plethora of variants exist in the literature [49]. It is the backbone of optimizing modern ML models, and has be proven incredibly effective in practice, especially enabling the rapid progress of AI and specifically Deep Learning [12]. However, as mentioned earlier, GD cannot be applied to arbitrary optimization problems and is only guaranteed to find a local optimum. In this thesis, we are concerned with optimization problems whose domain is discrete, where GD cannot be applied. A prominent example is ILP [36], where the domain is defined as $\mathbb{X} = \mathbb{Z}^n$ for some $n \in \mathbb{N}$ and we are faced with a linear objective. In the following section, we examine finite domains, as it is often the case in practice.

2.2.1 Combinatorial Optimization

CO [50] is a branch of optimization that deals with problems with the goal of finding the best solution from a finite but potentially vast set of possible solutions. The term "combinatorial" reflects that the problems often involve discrete structures such as sets, graphs, or sequences, and the solutions require combining elements in optimal ways. CO is omnipresent in practice and appears in numerous applications such as logistics [1], supply chains [51], scheduling [52], chip design [53], vehicle routing [54], energy systems [4] or healthcare [3]. In this thesis, we are mainly concerned with Quadratic Binary Programming (QBP) problems.

Definition 2.5 (QBP)**.** Let $\boldsymbol{W} \in \mathbb{R}^{n \times n}$, $\boldsymbol{A} \in \mathbb{Z}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{Z}^m$. QBP is defined as finding the solution to the following optimization problem

$$\min_{\boldsymbol{z} \in \mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{W} \boldsymbol{z} \tag{2.3a}$$

$$\text{s.t. } \boldsymbol{A}\boldsymbol{z} \preceq \boldsymbol{b} , \tag{2.3b}$$

We here use integer constraints, since it aligns with the natural combinatorial structure of many real-world problems modeled using QBP. Note that we do not allow the constraints to take arbitrary form, but we here restrict ourselves to linear constraints. Written in the form of Equation (2.2), our objective function becomes $f(\boldsymbol{z}) = \boldsymbol{z}^\top \boldsymbol{W} \boldsymbol{z}$ and the constraints $g_i(\boldsymbol{z}) = \boldsymbol{A}_i^\top \boldsymbol{z} - b_i$, where $\boldsymbol{A}_i^\top$ denotes the i-th row of $\boldsymbol{A}$.

The problem in Equation (2.3) may look rather specific but is actually a fairly general form of many

CO problems. In fact, all the aforementioned problems arising in real-world applications can be brought into the QBP form. Since current quantum computers cannot handle optimization under arbitrary constraints, we are interested in a more specific problem structure.

2.2.2 Quadratic Unconstrained Binary Optimization

QUBO [9] is a general purpose optimization format in CO. It is of central importance in this thesis due to its versatility and suitability for QC.

Definition 2.6 (QUBO)**.** Let $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$. QUBO aims to solve

$$\min_{\boldsymbol{z}\in\mathbb{B}^n} E_{\boldsymbol{Q}}(\boldsymbol{z}), \quad E_{\boldsymbol{Q}}(\boldsymbol{z}) \coloneqq \boldsymbol{z}^\top \boldsymbol{Q}\boldsymbol{z} = \sum_{i=1}^{n}\sum_{j=1}^{n} Q_{ij} z_i z_j \ . \tag{2.4}$$

We call $E_{\boldsymbol{Q}}$ the QUBO energy function.

Similarly to QBP, QUBO is highly versatile and can be used to model various optimization problems [55]. Examples range from satisfiability [56] over routing [5] and resource allocation [57] to computer vision [41] and ML [43, 58–60].

Note that any QUBO problem is fully characterized by a square matrix $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$, which we denote as *parameter matrix*. Due to the symmetries in the quadratic form $\boldsymbol{z}^\top \boldsymbol{Q}\boldsymbol{z}$, one often stumbles across alternative definitions of QUBO in the literature, with all of them being equivalent. For example, it is often assumed that $\boldsymbol{Q}$ is symmetric and that we are also given an offset vector $\boldsymbol{q} \in \mathbb{R}^n$ with the goal of solving

$$\min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{Q}\boldsymbol{z} + \boldsymbol{q}^\top \boldsymbol{z} \ .$$

Since $z_i z_i = z_i$ for $z_i \in \mathbb{B}$, we can add $\boldsymbol{q}$ on the diagonal of $\boldsymbol{Q}$ ($\mathrm{diag}(\boldsymbol{Q}) \mapsto \mathrm{diag}(\boldsymbol{Q}) + \boldsymbol{q}$) to get rid of the linear offset. Due to the fact that $z_i z_j = z_j z_i$, the assumption of symmetry or any other structure on $\boldsymbol{Q}$ can be leveraged by setting $Q_{ij} \to Q_{ij} + Q_{ji}$ and $Q_{ji} \to 0$ for all $i < j$. This leads to an upper triangular QUBO matrix, which completely characterizes the QUBO and uniquely defines the energy function $E_{\boldsymbol{Q}}$. Thus, we regularly assume a QUBO matrix to be upper triangular and denote the set of all upper triangular matrices as $\mathcal{Q}_n \subseteq \mathbb{R}^{n\times n}$.

In most practical applications, one is not interested in the actual minimum energy value, but in the configuration with such a value.

Definition 2.7. Let $\boldsymbol{Q} \in \mathcal{Q}_n$. The set of optimal configurations of $E_{\boldsymbol{Q}}$ is given by

$$\mathcal{Z}^*(\boldsymbol{Q}) \coloneqq \arg\min_{\boldsymbol{z}\in\mathbb{B}^n} E_{\boldsymbol{Q}}(\boldsymbol{z}) = \left\{\boldsymbol{z} \in \mathbb{B}^n : E_{\boldsymbol{Q}}(\boldsymbol{z}) \leq E_{\boldsymbol{Q}}(\boldsymbol{z}')\ \forall \boldsymbol{z}' \in \mathbb{B}^n\right\} \ .$$

We denote $\boldsymbol{z} \in \mathcal{Z}^*(\boldsymbol{Q})$ as an *optimizer* of $\boldsymbol{Q}$.

Clearly, $\boldsymbol{Q}$ is not restricted to have a unique optimizer, i.e., $|\mathcal{Z}^*(\boldsymbol{Q})| > 1$.

Finding a Solution Even though QUBO problems are often used in practice, finding an optimal solution is NP-hard [61]. That is, finding an optimal solution can be as hard as trying out every possible candidate solution (brute-force). Since it holds that $|\mathbb{B}^n| = 2^n$, the number of possible solutions is

exponentially large, which often renders finding an exact optimizer impossible. Nonetheless, different methods for obtaining an optimal solution have been developed [9, 10], with an exponential runtime in the worst case. Hence, state-of-the-art methods are (meta)heuristics, which find local optima and converge towards an optimal solution. Among the most prominent methods are Simulated Annealing (SA) [62], tabu search [7] and genetic algorithms [8], while more sophisticated methods are given by commercial solvers, such as Gurobi [63] and CPLEX [64]. A detailed overview of solution techniques, encompassing both exact and approximate methods, is given in [65].

In practice, results obtained with the aforementioned classical methods are often not satisfactory. This can be either due to the solution quality or the time needed for finding a good solution. In Chapters 5 to 7, we will discuss three use-cases where this caveat can be very critical. QC is a prominent candidate to overcome classical limitations. The bridge between QUBO and QC is given by the *Ising model*.

2.2.3 Ising Model

The Ising model [66] is a computational model from statistical physics. It was originally designed to model the magnetic spins of atomic particles in a lattice, but can be used to model much wider range of problems [55].

Definition 2.8 (Ising)**.** Let $\boldsymbol{J} \in \mathbb{R}^{n \times n}$ be hollow ($\operatorname{diag}(\boldsymbol{J}) = \mathbf{0}$), $\boldsymbol{h} \in \mathbb{R}^n$ and $n \in \mathbb{N}$. The energy of a bipolar vector $\boldsymbol{\sigma} \in \mathbb{S}^n$ with $\mathbb{S}^n \coloneqq \{-1, +1\}$ of the Ising model is defined as

$$E_{\boldsymbol{J},\boldsymbol{h}}(\boldsymbol{\sigma}) = \boldsymbol{\sigma}^\top \boldsymbol{J} \boldsymbol{\sigma} + \boldsymbol{h}^\top \boldsymbol{\sigma} \ . \tag{2.5}$$

Hollowness is assumed since $\sigma_i^2 = 1$ for $\sigma_i \in \mathbb{S}$, which would just add a constant offset $\operatorname{diag}(\boldsymbol{J})^\top \mathbf{1}$ to Equation (2.5), not relevant for optimization.

From a physics-viewpoint, the variable σ_i represents the magnetic spin of a lattice node i, J_{ij} the interaction between node i and j and h_i an external magnetic field acting on node i. Similar to the definition of QUBO in the literature, there exist different conventions for formalizing the Ising model. However, they are all equivalent to the one given in Equation (2.5).

Similar to Definition 2.7, we define $\mathcal{Z}^*(\boldsymbol{J}, \boldsymbol{h}) \coloneqq \arg\min_{\boldsymbol{\sigma} \in \mathbb{S}^n} E_{\boldsymbol{J},\boldsymbol{h}}$. We show the equivalence between the QUBO energy and the Ising energy.

Proposition 2.1. *Given* $\boldsymbol{Q} \in \mathbb{R}^{n \times n}$*, let* $\boldsymbol{J} = \frac{1}{4}\boldsymbol{Q}$*,* $\boldsymbol{h} = \frac{1}{2}\boldsymbol{Q}\mathbf{1}$*,* $c = \frac{1}{4}\mathbf{1}^\top \boldsymbol{Q}\mathbf{1}$ *and* $\iota : \mathbb{S}^n \to \mathbb{B}^n$*,* $\iota(\boldsymbol{\sigma}) = \frac{1}{2}(\mathbf{1} + \boldsymbol{\sigma})$*. Then*

$$E_{\boldsymbol{Q}}(\iota(\boldsymbol{\sigma})) = E_{\boldsymbol{J},\boldsymbol{h}}(\boldsymbol{\sigma}) + c, \ \forall \boldsymbol{\sigma} \in \mathbb{S}^n \ . \tag{2.6}$$

Specifically, the restriction $\iota|_{\mathcal{Z}^*(\boldsymbol{J},\boldsymbol{h})} : \mathcal{Z}^*(\boldsymbol{J}, \boldsymbol{h}) \to \mathcal{Z}^*(\boldsymbol{Q})$ *is an isomorphism.*

Proof. We use the the definition in Equation (2.4) to obtain

$$\begin{aligned} E_{\boldsymbol{Q}}\left(\iota(\boldsymbol{\sigma})\right) &= \frac{1}{4}(\mathbf{1}+\boldsymbol{\sigma})^{\top}\boldsymbol{Q}(\mathbf{1}+\boldsymbol{\sigma}) \\ &= \frac{1}{4}\left(\mathbf{1}^{\top}\boldsymbol{Q}\mathbf{1}+\mathbf{1}^{\top}\boldsymbol{Q}\boldsymbol{\sigma}+\boldsymbol{\sigma}^{\top}\boldsymbol{Q}\mathbf{1}+\boldsymbol{\sigma}^{\top}\boldsymbol{Q}\boldsymbol{\sigma}\right) \\ &= \frac{1}{4}\boldsymbol{\sigma}^{\top}\boldsymbol{Q}\boldsymbol{\sigma}+\frac{1}{2}\left(\boldsymbol{Q}\mathbf{1}\right)^{\top}\boldsymbol{\sigma}+\frac{1}{4}\mathbf{1}^{\top}\boldsymbol{Q}\mathbf{1} \\ &= \boldsymbol{\sigma}^{\top}\boldsymbol{J}\boldsymbol{\sigma}+\boldsymbol{h}^{\top}\boldsymbol{\sigma}+c \\ &= E_{\boldsymbol{J},\boldsymbol{h}}(\boldsymbol{\sigma})+c\,. \end{aligned}$$

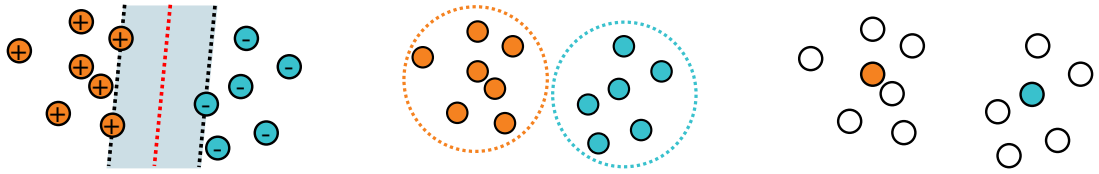
□

Thus, to find an optimum to a QUBO problem with matrix $\boldsymbol{Q}$, we can optimize the Ising model given in Proposition 2.1, with the reverse also holding true. However, solving an Ising model is as hard as solving QUBO, which is NP-hard. Hardware solvers aiming to find solutions for Ising models and equvialent QUBO formulations are called *Ising machines*. They are designed to solve CO problems by mimicking physical processes of magnetic systems. Many different approaches constructing Ising machines exist, encompassing analog, digital and quantum variants [67]. For the analog computation approach, numerous physical implementations for thermal annealing exist, e.g., including magnetic devices [68] and Bose-Einstein condensates [69]. Examples based on digital hardware are implementations of SA and evolutionary algorithms on Application-Specific Integrated Circuits (ASICs) [70], FPGAs [71, 72], or Graphical Processing Units (GPUs) [73, 74]. Dynamical-system solvers implemented with optics and electronics, such as coherent Ising machines [75, 76], also attracted interest lately. QC technologies have gained a lot of attention over the recent years for tackling QUBO problems. The most prominent example is Quantum Annealing (QA), which is a physical implementation of AQC. Through the exploitation of quantum tunneling, the system is allowed to pass through energy barriers and find good solutions of the underlying optimization problem. A more detailed elaboration is given in Section 2.4.3. All of these approaches have the advantages of being standalone without additional overhead and being able to massively parallelize computations.

As we have seen before, optimization is of great importance for many real-world tasks. It is especially vital for ML as it serves as the fundamental mechanism that enables models to learn from data.

2.3 Machine Learning

For many complex tasks, precise mathematical models or exact algorithms are challenging to develop. These tasks often demand a level of intelligence akin to human capabilities, such as recognizing shapes and objects in images, identifying patterns in data, or making predictions based on prior observations. Numerous methods have been devised to address these challenges, giving rise to the field of ML. ML is a subset of the broader field of AI, setting itself apart from traditional problem-solving approaches where machines operate strictly according to predefined rules. It is particularly valuable for tasks like pattern recognition, which are intuitive for humans but lack easily definable rules, or for problems where simple rules exist but the sheer complexity renders rule-based methods computationally impractical. In recent decades, advancements in hardware have significantly enhanced ML's relevance, making it a cornerstone technology across a broad spectrum of applications.



(a) Given labeled data, find maximum separating margin.

(b) Given unlabeled data, find similar groups.

(c) Given unlabeled data, find representative points.

Figure 2.1: Schematic representation of three different ML tasks: SVM (a), clustering (b) and VQ (c). See Sections 2.3.1 and 2.3.2 for more details.

ML can be considered to be the science of fitting data to mathematical models. This process can be described with the help of the following terms:

- **Data set** $\mathcal{D}$: Finite sample of underlying unknown data distribution $\mathbb{D}$,
- **Model** $h_{\boldsymbol{\theta}}$: Parametrized mathematical model, able to describe a certain hypothesis,
- **Loss function** $\mathcal{L}$: Rates the quality of the current model .

Assume we are given a *dataset* $\mathcal{D}$, we want to find a suitable *model* $h_{\boldsymbol{\theta}}$ describing our data. The chosen model $h_{\boldsymbol{\theta}}$ stems from some hypothesis space, which is believed to have the ability to accurately characterize the underlying data distribution. Finding optimal parameters $\boldsymbol{\theta}$ is done by optimizing a *loss function* $\mathcal{L}$ w.r.t. the data and the model

$$\min_{\theta} \mathcal{L}(h_{\boldsymbol{\theta}}; \mathcal{D}) \ . \tag{2.7}$$

Three main paradigms arise in ML: SL (Section 2.3.2), UL (Section 2.3.2) and RL (Section 2.3.3).

2.3.1 Supervised Learning

SL aims to learn the mapping between inputs and their corresponding outputs. We want to find a function $h_{\boldsymbol{\theta}} : \mathbb{X} \to \mathbb{Y}$ which is optimal in terms of a loss function $\mathcal{L} : \mathbb{Y} \times \mathbb{Y} \to \mathbb{R}$. Seeing $\mathbb{X} \times \mathbb{Y}$ as a random variable with the joint distribution $\mathbb{D}$, we aim for minimizing the *expected loss*

$$\min_{\theta} \mathbb{E}_{\mathbb{D}} \left(\mathcal{L}(h_{\boldsymbol{\theta}}(\mathbb{X}), \mathbb{Y}) \right) \ .$$

In practice, we have no knowledge on the underlying distribution but only access to a finite sample—the dataset—consisting of input-output pairs $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i \in [n]} \subseteq \mathbb{X} \times \mathbb{Y}$. That is, we fit the parameterized model $h_{\boldsymbol{\theta}}$ to the given data by minimizing the *empirical loss*

$$\min_{\theta} \sum_{i \in [n]} \mathcal{L}(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) \ .$$

The input space $\mathbb{X}$ is called feature space and is typically modeled as a metric space (e.g., $\mathbb{R}^d$), where inputs might represent diverse data types such as images, texts, emails, gene sequences, networks, financial time series, or demographic data. Outputs can take various forms: they may be quantitative,

such as a temperature measurement or the concentration of a specific substance in the body, or qualitative/categorical, such as binary values or multi-class labels. The first type of problem, where the output consists of quantitative values, is typically referred to as *regression*, while the latter, involving qualitative or categorical outputs, is known as *classification*. Exemplary regression problems are the prediction of a house price based on features like size and location or estimating the temperature given current weather conditions. For classification, examples would be to identify a an email as spam or classifying images of animals into categories like "cat" or "dog".

Based on the task at hand, we require a parametric model that can be trained to solve the problem. Popular models include ANNs, *Decision Trees*, *Random Forests* and SVMs.

Support Vector Machines SVMs [30, 77] are among the most extensively studied ML models due to their effectiveness in both classification and regression tasks. SVMs are particularly known for their ability to handle high-dimensional data and find an optimal decision boundary that maximizes the margin between different classes, making them powerful for tasks with clear class separations. Their theoretical foundations, along with various extensions and kernels, have made SVMs a central topic in ML research and applications.

In its original form, an SVM is a binary classifier which takes a labeled dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i\in[n]}$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$ for $i \in [n]$, and separates them with a hyperplane . As there may be infinitely many such hyperplanes, an additional objective is to maximize the *margin*, which is the area around the hyperplane containing no data points, in order to obtain best generalization. A schematic representation of an SVM is given in Figure 2.1(a).

The hyperplane is represented as a normal vector $\boldsymbol{w} \in \mathbb{R}^d$ and an offset or *bias* $b \in \mathbb{R}$. To ensure correctness, every data point must lie on the correct side of the plane

$$(\boldsymbol{w}^\top \boldsymbol{x}_i - b)y_i \geq 1 \ ,$$

A feature vector which lies exactly on the border of the margin is called Support Vector (SV), i.e., a vector $\boldsymbol{x}_i$ with $(w^\top \boldsymbol{x}_i - b)y_i = 1$. In addition to correct classification, we also want to maximize the margin width. Assume $\boldsymbol{x}_+$ and $\boldsymbol{x}_-$ are SVs with labels $+1$ and -1, respectively. The margin width is then given by the projection of $\boldsymbol{x}_+ - \boldsymbol{x}_-$ on the unit normal vector of the hyperplane

$$\frac{1}{\|\boldsymbol{w}\|}\boldsymbol{w}^\top \left(\boldsymbol{x}^+ - \boldsymbol{x}^-\right) = \frac{1}{\|\boldsymbol{w}\|}\left(b + 1 - (b-1)\right) = \frac{2}{\|\boldsymbol{w}\|} \ .$$

As perfect linear separability for real-world data is unlikely, *slack variables* $\xi_i > 0$ allow for slight violations. The new correctness constraint is

$$(\boldsymbol{w}^\top \boldsymbol{x}_i - b)y_i \geq 1 - \xi_i \ ,$$

where we want to minimize the sum of all slack variables $\mathbf{1}^\top \boldsymbol{\xi}$.

Definition 2.9 (Primal SVM)**.** Given a labeled dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i\in[n]} \subset \mathbb{R}^d \times \mathbb{S}$, the primal SVM

Table 2.1: Overview of frequently used kernel functions [79].

Name	Form	Parameters
Linear	$\boldsymbol{x}^\top \boldsymbol{x}'$	
Polynomial	$(\boldsymbol{x}^\top \boldsymbol{x}' + c)^p$	Degree $p \in \mathbb{N}$, constant $c \in \mathbb{R}$
RBF/Gaussian	$\exp(-\gamma \lVert \boldsymbol{x} - \boldsymbol{x}' \rVert^2)$	Bandwidth $\gamma \in \mathbb{R}$

aims to optimize

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \lVert \boldsymbol{w} \rVert^2 + C\mathbf{1}^\top \boldsymbol{\xi} \tag{2.8a}$$

$$\text{s.t. } (\boldsymbol{w}^\top \boldsymbol{x}_i - b) \cdot y_i \geq 1 - \xi_i,\ \forall i \in [n]\ , \tag{2.8b}$$

where the parameter $C > 0$ determines the trade-off between increasing the margin size and ensuring that the $\boldsymbol{x}_i$ lie on the correct side of the margin.

Kernel Trick Many data distributions are not linear separable in the original space, or do not even live in a properly defined Hilbert space in the first place. In such cases, a feature map $\phi : \mathbb{X} \to \mathbb{R}^{d'}$ can be applied to project the data into a feature space where linear separation becomes feasible. However, explicitly computing the feature map can be computationally expensive, especially when the dimensionality of the feature space is much larger than the one of input space ($d \ll d'$). Furthermore, solving the objective in Equation (2.8), we need to compute d'^2 inner products of size n and invert a $d' \times d'$ matrix. Finding a solution of the primal SVM thus has a complexity of $\mathcal{O}(nd'^2 + d'^3)$, which is infeasible for large d'.

The dual formulation of a SVM addresses this challenge through the kernel trick, which avoids operating directly in the feature space. It is based on *Mercer's theorem* [78].

Theorem 2.1 (Mercer)**.** *Given a dataset $\mathcal{D} = \{\boldsymbol{x}_i\}_{i\in[n]} \subset \mathbb{X}$ and a symmetric function $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$. Let $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ with $K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. If $\boldsymbol{K}$ is positive semi-definite, then there exists a map $\phi : \mathbb{X} \to \mathbb{R}^D$, s.t.,*

$$K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j)\ .$$

The actual theorem is more general, which is applicable to square-integrable functions, rather than to the restriction of a finite dataset. A symmetric function $K(\cdot,\cdot)$ which fulfills the positive semi-definiteness condition is called Mercer kernel or just kernel in short. The corresponding matrix $\boldsymbol{K}$ is called kernel matrix.

The kernel trick is now to replace inner products of feature vectors by a suitable kernel. This approach enables efficient computation of inner products in the high-dimensional feature space without explicitly realizing the feature map, significantly reducing the computational overhead. An overview of commonly used kernel functions is given in Table 2.1.

Kernels can be applied to SVMs by considering its well-established *Lagrangian dual* form.

Definition 2.10 (Dual SVM)**.** Given a labeled dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i\in[n]} \subset \mathbb{X} \times \mathbb{S}$, we let $\boldsymbol{y} = (y_1, \ldots, y_n)^\top$ and $K : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ be a kernel function with corresponding kernel matrix $\boldsymbol{K} \in \mathbb{R}^{n\times n}$

respective the given dataset. The dual SVM objective is given by

$$\max_{\boldsymbol{\alpha}} \mathbf{1}^\top \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\top \left(\boldsymbol{y}\boldsymbol{y}^\top \odot \boldsymbol{K}\right) \boldsymbol{\alpha} \tag{2.9a}$$

$$\text{s.t. } \mathbf{0} \preceq \boldsymbol{\alpha} \preceq C\mathbf{1}, \tag{2.9b}$$

$$\boldsymbol{y}^\top \boldsymbol{\alpha} = 0\ . \tag{2.9c}$$

For solving Equation (2.9), we evaluate the kernel n^2 times and invert the corresponding $n \times n$ kernel matrix. Since the evaluation of the kernel is often possible in $\mathcal{O}(d)$ (c.f. Table 2.1), we obtain a total complexity of $\mathcal{O}(dn^2 + n^3)$. This can largely improve upon the complexity of the primal form, especially when the feature map is intractable to compute.

2.3.2 Unsupervised Learning

UL does not require the creator to "supervise" the model during training, enabling it to independently identify patterns and structures in data. Unlike SL, it operates on unlabeled datasets, i. e., $\mathcal{D} = \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n\} \subseteq \mathbb{X}$, making it well-suited for discovering hidden relationships. Prominent examples are *cluster analysis*, VQ and *anomaly detection*, where we want to detect "unusual" points in a dataset, which may be of special interest.

Clustering Clustering is an unsupervised ML technique used to group data points into clusters, where data points within the same cluster are more similar to each other than to those in other clusters. The goal of clustering is to identify patterns, structures, or relationships in data without the need for labeled outputs. A schematic representation is given in Figure 2.1(b).

A distinction is made between hard and soft clustering, where the first describes exclusive membership in clusters while the latter allows the membership to a certain degree. We focus on hard clustering, i. e., we want to find disjoint subsets $C_i \subset \mathcal{D}$ with $\bigcup_i C_i = \mathcal{D}$ such that points in C_i are similar and points from two different clusters C_i and C_j are dissimilar. Hence, one relies on suitable similarity measures such as Euclidean distance or cosine similarity. Many different methods exist all with certain advantages and disadvantages, e. g., *DBSCAN*, *hierarchical clustering* or *spectral clustering*. However, we here focus on the arguable most prominent approaches, *k-means clustering* and *k-medoids clustering*. For setting up the exact objective, we first define the mean/medoid of a cluster.

Definition 2.11 (Mean, Medoid)**.** Let $\mathcal{D} \subset \mathbb{R}^d$ and $C \subset \mathcal{D}$. The mean $\boldsymbol{\mu}$ of C is defined as

$$\boldsymbol{\mu} = \frac{1}{|C|} \sum_{\boldsymbol{x} \in C} \boldsymbol{x}\ ,$$

while the medoid $\boldsymbol{m}$ is defined to be an element of C

$$\boldsymbol{m} = \arg\min_{\boldsymbol{x}' \in C} \sum_{\boldsymbol{x} \in C} \left\|\boldsymbol{x} - \boldsymbol{x}'\right\|^2\ .$$

Note that if $\boldsymbol{\mu} \in C$, then $\boldsymbol{\mu} = \boldsymbol{m}$.

Definition 2.12 (***k***-Means, ***k***-Medoids)**.** Let $\mathcal{D} \subset \mathbb{R}^d$, $|\mathcal{D}| = n$, $C_i \subset \mathcal{D}$, $i \in [k]$, $k < n$ and

$$C_i \cap C_j = \emptyset,\ \forall i \neq j, \quad \bigcup_{i \in [k]} C_i = \mathcal{D}\ .$$

The objective of k-means/k-medoids clustering is given by minimizing the *within cluster scatter*

$$\min_{C_1,\dots,C_k} \sum_{i \in [k]} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2\ , \quad \min_{C_1,\dots,C_k} \sum_{i \in [k]} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{m}_i\|^2\ . \tag{2.10}$$

A notable characteristic of medoids is that they are determined solely by evaluating squared distances between given data points. Such distances can be precomputed and thus do not rely on numeric data, in opposition to means.

Finding a good solution to the objectives in Equation (2.10) is conceptually very similar. For example, it can be achieved by using slightly different versions of Lloyd's algorithm [80]. This method initializes k means/medoids and iteratively determines clusters by assigning data points to their closest mean/medoid and updates these means/medoids according to that assignment.

Vector Quantization VQ is a technique used in signal processing and ML to compress a dataset into a finite set of representative points, known as prototypes. For $\mathcal{D} = \{\boldsymbol{x}^1, \dots, \boldsymbol{x}^n\} \subset \mathbb{X}$, it aims to identify $k \ll n$ prototypes $\mathcal{W} = \{\boldsymbol{w}^1, \dots, \boldsymbol{w}^k\} \subset \mathbb{X}$ that serve as a compressed representation of the dataset. These representative points form a codebook, and each input vector is mapped to its closest prototype, effectively quantizing the data. A schematic representation is given in Figure 2.1(c).

It functions by partitioning a large set of points (vectors) into groups, with each group containing roughly the same number of points nearest to it. This makes VQ suitable for lossy data compression, pattern recognition and density estimation. Furthermore, it is very similar to clustering but describes a slightly different task. For example, one can use the cluster means $\boldsymbol{\mu}_i$ of k-means as prototypes for VQ.

As VQ aims to model the underlying probability density functions of the data distribution with the help of the codebook, we consider probability density estimates. That is, assuming that $p_{\mathcal{D}}(\cdot)$ is an underlying probability density function of $\mathcal{D}$ and $p_{\mathcal{W}}(\cdot)$ of $\mathcal{W}$, we want $p_{\mathcal{D}}(\cdot)$ and $p_{\mathcal{W}}(\cdot)$ to be as similar as possible. The dissimilarity or *divergence* can be measured in numerous ways, examples include the *Cauchy-Schwartz* divergence [81] or the *Kullback-Leibler* divergence [82].

Since we have no knowledge on the underlying distribution of our data, we use approximations. In particular, we examine Kernel Density Estimation (KDE), also called *Parzen windowing*, similar to [83].

Definition 2.13 (KDE)**.** Let $K : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ be a kernel function, $\mathcal{D} \subset \mathbb{X}$, $|\mathcal{D}| = n$. A KDE of the underlying distribution of $\mathcal{D}$ is given by

$$p_{\mathcal{D}}(\boldsymbol{x}) = \frac{1}{n} \sum_{\boldsymbol{x}' \in \mathcal{D}} K(\boldsymbol{x}, \boldsymbol{x}')\ .$$

In [84, 85], a Gaussian kernel is used to approximate $p_{\mathcal{D}}(\cdot)$ and $p_{\mathcal{W}}(\cdot)$ and the Cauchy-Schwartz divergence is minimized. However, we are not restricted in the choice of the used kernel function. Due

to Mercer's theorem (Theorem 2.1), there exists a feature map $\phi : \mathbb{X} \to \mathbb{R}^D$, s.t.,

$$p_{\mathcal{D}}(\boldsymbol{x}) = \frac{1}{n} \sum_{\boldsymbol{x}' \in \mathcal{D}} \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}') = \phi(\boldsymbol{x})^\top \boldsymbol{\phi}_{\mathcal{X}}, \quad \boldsymbol{\phi}_{\mathcal{D}} \coloneqq \frac{1}{n} \sum_{\boldsymbol{x} \in \mathcal{D}} \phi(\boldsymbol{x}), \tag{2.11a}$$

$$p_{\mathcal{W}}(\boldsymbol{x}) = \frac{1}{k} \sum_{\boldsymbol{x}' \in \mathcal{W}} \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}') = \phi(\boldsymbol{x})^\top \boldsymbol{\phi}_{\mathcal{W}}, \quad \boldsymbol{\phi}_{\mathcal{W}} \coloneqq \frac{1}{k} \sum_{\boldsymbol{x} \in \mathcal{W}} \phi(\boldsymbol{x}) \ . \tag{2.11b}$$

The mean vectors $\boldsymbol{\phi}_{\mathcal{X}}$ and $\boldsymbol{\phi}_{\mathcal{W}}$ fully characterize the KDEs $p_{\mathcal{D}}(\cdot)$ and $p_{\mathcal{W}}(\cdot)$. The difference between these vectors can be measured by the Mean Discrepancy (MD) [86], which leads to following objective.

Definition 2.14 (MD-VQ)**.** Let $K : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ be a kernel function, $\mathcal{D} \subset \mathbb{X}$, $|\mathcal{D}| = n$. MD-VQ aims to find a codebook $\mathcal{W} \subset \mathbb{X}$, $|\mathcal{W}| = k$ and $k \ll n$, s.t.,

$$\min_{\mathcal{W}} \ \|\boldsymbol{\phi}_{\mathcal{D}} - \boldsymbol{\phi}_{\mathcal{W}}\|^2 \tag{2.12a}$$

$$\text{s.t. } |\mathcal{W}| = k \ , \tag{2.12b}$$

where $\boldsymbol{\phi}_{\mathcal{X}}$ and $\boldsymbol{\phi}_{\mathcal{W}}$ are the feature mean vectors defined in Equation (2.11).

It is worth noting that prototypes coinciding with actual data points are often easier to interpret. This makes MD-VQ very similar to k-medoids, and we will investigate this connection later on in Chapter 3.

2.3.3 Reinforcement Learning

RL [87] is an ML paradigm in which an agent learns to make decisions by interacting with an environment to maximize a cumulative reward signal. Unlike SL, RL does not rely on labeled data; instead, it uses trial-and-error to discover optimal actions. The agent observes the current state of the environment, takes an action, and receives feedback in the form of a reward and a new state. Over time, it refines its behavior using strategies such as *Q-learning* [88] or *policy gradients* [89], balancing *exploration* (trying new actions) and *exploitation* (choosing the best-known actions). RL has applications in robotics, game playing [14], resource management, and other domains requiring sequential decision-making. We formalize this decision problem by a mathematical framework.

Markov Decision Process A decision process is a framework used to model decision-making in environments where outcomes are partly under the control of a decision-maker and partly under the influence of chance. One common formalization of a decision process is an MDP [90]. We will focus on deterministic environments, where we have full control over the outcomes. It can be described by a 4-tuple $(\mathcal{S}, \mathcal{A}, f, r)$:

- **State space** $\mathcal{S}$: Set of all possible states the system can be in. Each state provides a complete description of the system at a particular point in time.
- **Action space** $\mathcal{A}$: Set of all possible actions that the decision-maker can take. Actions are choices or moves that can influence the state of the system. It may be dependend on the current state.
- **Transition function** $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$: Function defining the transition of moving to a new state $s' \in \mathcal{S}$ given the current state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$, $f(s, a) = s'$. It encapsulates the dynamics of the environment.

- **Reward function** $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$: Function assigning a numerical value (reward) received after taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$. This value represents the immediate benefit or cost of performing the action in that state.

Note that the transitions can happen under uncertainty, as it is often the case in practice. Also, we often cannot fully observe our current state, which leads to the formalization of a *partially observable* MDP.

The next decision in the current state is governed by a policy $\pi : \mathcal{S} \to \mathcal{A}$ which maps the current state to the action to be taken next. Similarly to uncertain transitions, policies are often described by probability densities in practice, i. e., with which probability should we follow action a starting in state s. Rating the quality of a policy π is done by having a look at the gathered cumulative reward following π. It can be described by the *value function*.

Definition 2.15 (Value function)**.** Let $(\mathcal{S}, \mathcal{A}, f, r)$ be an MDP and $\gamma \in (0, 1)$. The value function $V^\pi : \mathcal{S} \to \mathbb{R}$ is defined as

$$V^\pi(s) \coloneqq \sum_{t=0}^{T} \gamma^t r\left(s_t, \pi(s_t)\right), \quad s_0 = s\ , \tag{2.13}$$

where $\gamma \in (0, 1)$ is a discount factor guaranteeing convergence for infinite horizons $T \to \infty$ and the state evolution is described by $s_{t+1} = f\left(s_t, \pi(s_t)\right)$.

The function V^π rates how good a policy π performs, that is, the higher the cumulative reward the better. As we want to maximize our rewards, the goal of an MDP is to find an optimal policy π^* which maximizes Equation (2.13), i. e.,

$$\pi^*(s) = \arg\max_{\pi:\mathbb{S}\to\mathbb{A}} V^\pi(s), \quad \forall s \in \mathcal{S}\ . \tag{2.14}$$

In practice, one also wants to rate the quality of a taken action.

Definition 2.16 (Action value)**.** Let $(\mathcal{S}, \mathcal{A}, f, r)$ be an MDP and $\gamma \in (0, 1)$. Define the Q*-value* or *action value* as

$$Q^\pi(s, a) \coloneqq r(s, a) + \gamma V^\pi(f(s, a))\ ,$$

which describes the quality of a state-action pair.

In order to solve Equation (2.14), one often utilizes a fundamental recursive insight.

Theorem 2.2 (Bellman [91])**.** *Let $(\mathcal{S}, \mathcal{A}, f, r)$ be an MDP and $\gamma \in (0, 1)$. The* Bellman equation *is a recursive description of the value function*

$$V^\pi(s) = \max_{a \in \mathcal{A}(s)} r(s, a) + \gamma V^\pi(f(s, a)), \quad \forall s \in \mathcal{S}\ . \tag{2.15}$$

This also leads to a recursive description of the Q-value using the Bellman equation

$$Q^\pi(s, a) = r(s, a) + \gamma \max_{a \in \mathcal{A}(s)} Q^\pi(f(s, a), a)\ . \tag{2.16}$$

One approach for finding π^* is value iteration: Instead of optimizing directly over possible policies, the value function (or Q-value function) is iteratively updated until convergence. Let us denote the

corresponding approximations of V^{π^*} (or Q^{π^*}) with $\dot{V}$ (or $\dot{Q}$). The resulting policy $\dot{\pi}$ can then be constructed by

$$\dot{\pi}(s) = \arg\max_{a \in \mathcal{A}(s)} \dot{Q}(s,a), \quad \forall s \in \mathcal{S} \; . \tag{2.17}$$

There are many different ways of obtaining an approximately optimal Q-value function $\tilde{Q}$.

Q-Learning One of the most prominent examples for discrete action spaces is Q-learning [88]. The Q-function $\dot{Q}$ is initialized before the learning starts and then the following update is conducted iteratively:

$$\dot{Q}(s,a) \leftarrow (1-\alpha)\dot{Q}(s,a) + \alpha \bar{Q}(s,a), \; \bar{Q}(s,a) = r(s,a) + \gamma \max_{a \in \mathbb{A}(s)} \dot{Q}(s,a) \; .$$

The learning rate $0 < \alpha \leq 1$ weights the current value $\dot{Q}(s,a)$ with the new information $\bar{Q}(s,a)$, which corresponds to applying the Bellman equation Equation (2.16).

For discrete state and action spaces, one often uses tabular storage to keep track of $\dot{Q}(s,a)$ for each state-action pair. However, if the state space is continuous, this method is no longer feasible. In those cases, one falls back to function approximation of $\dot{Q}$ instead of storing all evaluations of $\dot{Q}$. The most prominent function approximators are of course ANNs. The combination of Q-learning with using an ANN to represent $\dot{Q}$ is called deep Q-learning [92].

Policy Rollout A suboptimal policy $\dot{\pi}$ can further be improved by using the concept of Policy Rollout (PR) [93, 94]. Obtaining $\dot{\pi}$ can for example be obtained as a myopic greedy policy

$$\dot{\pi}(s) = \arg\min_{a \in \mathcal{A}} \; r(s,a) \; , \tag{2.18}$$

or by using Q-learning. PR describes the concept of following such a given base policy $\dot{\pi}$ for a certain number of steps. Its usage is motivated by the well known *rollout selection policy*, which approximates the optimal future reward $V^{\pi^*}(f(s_t,a))$ with $V^{\dot{\pi}}(f(s_t,a))$. The following approximation of the Bellman equation Equation (2.15) is used

$$V^{\tilde{\pi}}(s_t) = \max_{a \in \mathcal{A}} \left[r(s_t,a) + V^{\dot{\pi}}(f(s_t,a)) \right] \; . \tag{2.19}$$

The rollout selection policy $\tilde{\pi}$ is guaranteed to be at least as good as the base policy $\dot{\pi}$ and often outperforms it [93]. Its simplicity and strong performance stem from its close connection to the core dynamic programming algorithm of policy iteration. Equation (2.19) describes the optimal one-step look-ahead policy, which is followed by adhering to the base policy thereafter.

There are several variants of the rollout selection policy for reducing computational requirements. *Simplified rollout* is employed to reduce the action space size in every step by choosing a subset due to some measure. This bears some resemblance to the *beam search* method [95] for exploration, where a fixed-size search beam is maintained for traversing through the search space. Another way of reducing computational burden is to use *truncated rollout*, which stops the PR before reaching the full horizon.

2.4 Quantum Computing

QC [19] leverages the principles of quantum mechanics, such as *superposition*, *entanglement*, and *quantum tunneling*, to process information in fundamentally different ways compared to classical computing. Recently, it has gained widespread attention due to the upcoming availability of quantum hardware through companies like *IBM Quantum* and *D-Wave*. For more details on the availability and current state of QC hardware, we refer to Section 2.4.4.

In theory, QC is able to solve certain tasks way faster than classical computers [22, 23]. A practical quantum hardware demonstration of solving a problem infeasible for any classical computer within a reasonable timeframe, is called *quantum supremacy* or *quantum advantage*[1]. Google presented a remarkable sampling benchmark performance of their newest quantum chip in December 2024, performing computation in under 5 minutes, while the fastest supercomputers would need around 10^{25} years[2] for the same task. D-Wave claimed quantum supremacy by using their *Advantage Systems* for simulating non-equilibrium dynamics in March 2025 [96] and for complex magnetic materials simulation problems with relevance to materials discovery in March 2025 [97].

Even though the usefulness of such sampling and simulation benchmarks can be called into question, the promising theoretical properties of QC sparked a plethora of research investigating its potential for different applications. Two application areas gained particular wide-spread attention in the last decade—QML and QO. QML seeks to leverage QC techniques for common ML tasks, including classification, regression, and clustering [26, 79], while QO focuses on potential benefits for optimization. A more in-depth description of QO is given in Section 2.4.3.

Qubits At the heart of QC is the concept of replacing classical bits with quantum bits, called *qubits*. While classical bits can only have distinct values in $\mathbb{B} = \{0, 1\}$, that is, either the value 0 or 1, qubits exist in continuous states "between" 0 and 1 (superposition) and yield discrete values only when measured.

One denotes the state of qubits in *Bra-ket notation*, stemming from the physics area of quantum mechanics. That is, we say a single qubit is in state $|\psi\rangle$ (ket-vector), which is a two-dimensional complex vector

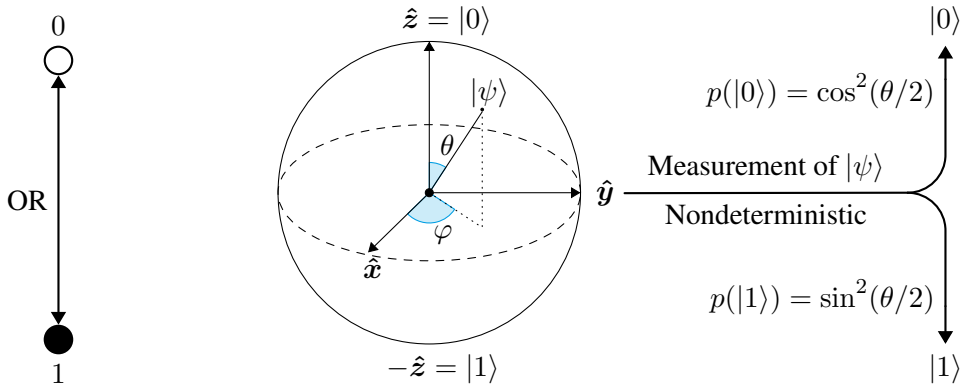
$$|\psi\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} = \alpha_0 |0\rangle + \alpha_1 |1\rangle \in \mathbb{C}^2, \; |0\rangle := \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \; |1\rangle := \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Similar to classical bits, qubits have two *basis states* $|0\rangle$ and $|1\rangle$ which coincide with the standard basis vectors in two dimensions. A single qubit can be in a mixture of states and if $\alpha_0 \neq 0$ and $\alpha_1 \neq 0$, we say the qubit is in superposition. The bra vector $\langle\psi|$ denotes the conjugate transpose of $|\psi\rangle$, that is $\langle\psi| = (|\psi\rangle)^\dagger = (\alpha_0^*, \alpha_1^*)$, where * indicates complex conjugation. We assume $\mathbb{C}^2$ as the Hilbert space equipped with the standard inner product in complex spaces, that is $\langle\phi|\psi\rangle := \langle\phi| \, |\psi\rangle$. The complex coefficients α_0 and α_1 are called the amplitudes and the amplitude vector $|\psi\rangle$ is assumed to be normalized

$$\langle\psi|\psi\rangle = |\alpha_0|^2 + |\alpha_1|^2 = 1.$$

Since quantum states are normalized, the state of a single qubit can be seen as a vector on a unit sphere—the so-called *Bloch* sphere. Due to the loss of a degree of freedom through the normalization

[1] https://www.quantamagazine.org/john-preskill-explains-quantum-supremacy-20191002 (last accessed September 19, 2025)
[2] https://blog.google/technology/research/google-willow-quantum-chip (last accessed September 19, 2025)



(a) Classical bit can only exist in one of two definite states, either 0 or 1.

(b) Qubits can exist in a superposition of their basis states $|0\rangle$ and $|1\rangle$ Equation (2.20). A clear definition of its state is given through measurement.

Figure 2.2: State comparison between a classical bit and a qubit.

assumption, we can also use an equivalent polar state description with azimuth angle ϕ and polar angle θ

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle \ . \tag{2.20}$$

A visualization of a single-qubit state along with a comparison to classical bits can be found in Figure 2.2.

Measurement Unfortunately, a quantum state cannot be directly observed, i. e., we do not have access to the amplitudes of a qubit. If we try to *measure* such a quantum state, it causes a collapse into one of its basis states. The probability of a qubit being measured in a particular basis state is given by the absolute square of the corresponding amplitude. The normalization property of qubit states guarantees that the probabilities of all basis states sum to 1, aligning with probability theory. During measurement, information on the quantum state gets lost, which gets clear through the following example. Looking at the equal superposition states

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \quad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \ ,$$

we realize, the measurement probability of ending up in either basis state is $\frac{1}{2}$ for both states. However, the states $|+\rangle$ and $|-\rangle$ are clearly not the same.

Quantum Register The real power of QC comes from combining states of multiple single qubits into one *quantum register*. We describe a system consisting of n qubit states $|\psi_1\rangle, \ldots, |\psi_n\rangle$ by a combined amplitude vector ψ

$$|\psi\rangle = |\psi_1 \ldots \psi_n\rangle = |\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle \ .$$

Note that the i-th entry of $|\psi\rangle$ is the amplitude α_i of the standard basis vector $|i\rangle := \boldsymbol{e}_i^N$, where $N = 2^n$. That is, $|\psi\rangle$ can be written as $|\psi\rangle = \sum_{i=0}^{N-1} \alpha_i |i\rangle$. Multi-qubit states which can be expressed by Kronecker products of single-qubit states are called *seperable*. That is, the qubits are *independent* and

such states can be efficiently simulated with classical computers. The true power of QC comes from quantum registers which are not decomposable. The simplest states of this form are the 2-qubit *Bell states*

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}},\ |\Phi^-\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}},\ |\Psi^+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}},\ |\Psi^-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}\ .$$

$|\Phi^+\rangle$, $|\Phi^-\rangle$ have an equal probability of $\frac{1}{2}$ to be measured in $|00\rangle$ or $|11\rangle$, while $|\Phi^+\rangle$, $|\Phi^-\rangle$ in $|01\rangle$ or $|10\rangle$. However, they cannot be written as a single Kronecker product of the basis states. After measurement, the outcome of the first qubit automatically determines the one from the second qubit and vice versa. Speaking from a probabilistic viewpoint, the probability distributions of the single qubits are not independent and thus do not factorize. This phenomenon is described as *entanglement* and has no analogue in our classical world. With full entanglement, the most efficient way of storing the amplitudes is one-by-one, which makes highly entangled quantum states intractable to simulate with a classical computer.

Observables So far, we assumed that we take a measurement in the standard basis, that is in the **Z**-basis. This is due to the fact that the Pauli-Z matrix (Table 2.2, more details will follow in the upcoming section) can be diagonalized in the standard basis

$$\mathbf{Z} = (+1)\,|0\rangle\,\langle 0| + (-1)\,|1\rangle\,\langle 1|\ ,$$

i.e., $\mathbf{Z}$ has eigenvalues $-1, +1$ and eigenvectors $|0\rangle, |1\rangle$. Similarly, we can think of a measurement procedure which does not give us the probabilities for the standard basis states $|0\rangle$ and $|1\rangle$, but for a different basis, such as $|+\rangle$ and $|-\rangle$. As is known from standard linear algebra theory, such a basis change can be achieved by finding a suitable unitary matrix. In the our exemplary case, we can achieve this by using the *Hadamard* matrix $\mathbf{H}$ (Table 2.2) on an arbitrary state $|\psi\rangle$ in the standard basis to obtain a state $|\psi\rangle'$ in the **X**-basis

$$|\psi\rangle' = \mathbf{H}\,|\psi\rangle = \mathbf{H}(\alpha_0\,|0\rangle + \alpha_1\,|1\rangle) = \alpha_0\mathbf{H}\,|0\rangle + \alpha_1\mathbf{H}\,|1\rangle = \alpha_0\,|+\rangle + \alpha_1\,|-\rangle\ .$$

Generalizing this concept to quantum registers, we ask the question whether we can also obtain different measurable properties. This is answered by an *observable* $\boldsymbol{O} \in \mathbb{C}^{2^n}$, which is a measurable physical quantity that can be assigned a numerical value, such as spin, position, momentum, or energy. Hence, the matrix $\boldsymbol{O}$ is often assumed to be Hermitian, since it then has real eigenvalues $\lambda_i \in \mathbb{R}$. With the aforementioned basis change, we assume that it can be decomposed to $\boldsymbol{O} = \sum_{i=0}^{N-1} \lambda_i\,|i\rangle\,\langle i|$. A measurement of a quantum register $|\psi\rangle$ w.r.t. an observable $\boldsymbol{O}$ then results in an eigenvalue λ_i and it causes the state to collapse into the *eigenstate* $|i\rangle$. The probability of measuring λ_i is

$$p(\lambda_i) = |\langle i|\psi\rangle|^2 = \sum_{j=0}^{N-1} \alpha_j\,|\langle i|j\rangle|^2 = |\alpha_i|^2\ .$$

Thus, measuring an eigenstate of some Hermitian observable always results in the same result. This means, that if the current state $|\psi\rangle$ is not in such an eigenstate, we cannot completely observe it and have a nondeterministic measurement outcome. In practice, one is often interested in the expectation

Table 2.2: Overview of frequently used quantum gates along with their associated unitaries.

Name	Symbol	#Qubits	Parameters	Unitary	Circuit
NOT/Pauli X	$\mathbf{X}$	1		$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	
Pauli Y	$\mathbf{Y}$	1		$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	
Pauli Z	$\mathbf{Z}$	1		$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	
Hadamard	$\mathbf{H}$	1		$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	
X-Rotation	$\mathbf{R_X}$	1	$\theta \in [0, 2\pi)$	$\begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$	
Y-Rotation	$\mathbf{R_Y}$	1	$\theta \in [0, 2\pi)$	$\begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$	
Z-Rotation	$\mathbf{R_Z}$	1	$\theta \in [0, 2\pi)$	$\begin{pmatrix} \exp\left(-i\frac{\theta}{2}\right) & 0 \\ 0 & \exp\left(i\frac{\theta}{2}\right) \end{pmatrix}$	
Controlled NOT	$\mathbf{CX}$	2		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	

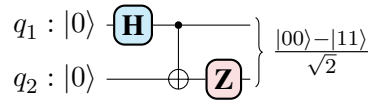
of an observable $\boldsymbol{O}$, i.e., $\langle \boldsymbol{O} \rangle = \langle \psi | \boldsymbol{O} | \psi \rangle$. As in classical statistics, it is approximated by taking many measurements and averaging the corresponding obtained results.

2.4.1 Quantum Gate Computing

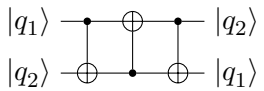
Having the necessary details at hand, we can go over to the most prominent paradigm of QC, namely QGC. It describes the framework of constructing and manipulating quantum states. This is done by *quantum circuits*, which are fundamental structures used to perform computations on a quantum computer. They consist of a sequence of *quantum gates* applied to a set of qubits, where each gate represents a specific quantum operation. These circuits are analogous to classical logic circuits, but operate under the principles of quantum mechanics.

Quantum Gates Quantum gates are represented by unitary matrices, which transform the amplitudes by multiplication with the given quantum state. A unitary matrix $\boldsymbol{U}$ fulfills the property $\boldsymbol{U}^\dagger = \boldsymbol{U}^{-1}$, which preserves the length of a vector after multiplication. Unitarity ensures the preservation of the normalization property $\langle \psi | \psi \rangle = 1$ and thus the reversibility of quantum computations, a core requirement in quantum mechanics. An overview of the most prominent quantum gates can be found in Table 2.2. The gates $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ are called *Pauli*-gates and are building blocks for more complex quantum gates. They are *Hermitian*, that is $\boldsymbol{U} = \boldsymbol{U}^\dagger$, and thus self-involutory ($\mathbf{XX} = \mathbf{YY} = \mathbf{ZZ} = \boldsymbol{I}$). For example, the $\mathbf{X}$-gate (*NOT* gate) flips the basis state of a single qubit

$$\mathbf{X}\,|0\rangle = |1\rangle\,, \quad \mathbf{X}\,|1\rangle = |0\rangle\ .$$



(a) Circuit creating the Bell state $|\Phi^-\rangle$.



(b) Circuit for swapping qubit states.

Figure 2.3: Exemplary 2-qubit quantum circuits.

The **H**-gate (*Hadamard* gate) has the effect of mapping a single-qubit basis state onto an equal superposition of the basis states

$$\mathbf{H}\,|0\rangle = |+\rangle\,, \quad \mathbf{H}\,|1\rangle = |-\rangle\ ,$$

while the gates $\mathbf{R_X}(\theta) = e^{-i\frac{\theta}{2}\mathbf{X}}$, $\mathbf{R_Y}(\theta) = e^{-i\frac{\theta}{2}\mathbf{Y}}$, $\mathbf{R_Z}(\theta) = e^{-i\frac{\theta}{2}\mathbf{Z}}$ act like a rotation on single qubits. From the Bloch sphere point-of-view, they rotate the state vector by θ along the x-, y- and z-axis, respectively. The most important two-qubit gate is the **CX**-gate (also called *controlled NOT* or **CNOT**) which flips the state of the first/target qubit when the second/control qubit is in the $|1\rangle$ state and leaves the target qubit unchanged otherwise. In quantum hardware, this gate is often used to create entanglements between single qubits.

Quantum Circuits A quantum circuit describes a sequence of initialization of qubits, quantum gates and measurements. It can be depicted graphically such that the horizontal axis represents time, progressing from left to right. Horizontal lines denote qubits, while double lines indicate classical bits. The elements connected by these lines correspond to operations performed on the qubits, such as gates or measurements. The underlying unitary $\boldsymbol{U}$ of the given quantum circuit can be written as a finite product of quantum gates $\{\boldsymbol{U}_1, \ldots, \boldsymbol{U}_m\}$, i.e., $\boldsymbol{U} = \boldsymbol{U}_m \cdots \boldsymbol{U}_1$. In this case, we say that the circuit has *depth* m, which is a key property used to describe the complexity of quantum algorithms. Note that the order of the gates can be confusing: we read the circuit diagram from left to right, that is the gates appear in ascending order of the indices. However, writing down the product of the gates, the first gate to be applied is the last factor, since we multiply from the left. Exemplary quantum circuits are depicted in Figures 2.3 to 2.5.

A set of universal quantum gates is a collection of gates capable of represented any operation on a quantum computer. In other words, any unitary matrix $\boldsymbol{U}$ can be approximated to an arbitrary precision $\epsilon > 0$ by a finite sequence $\{\boldsymbol{U}_1, \ldots, \boldsymbol{U}_m\}$ of gates from this set

$$\|\boldsymbol{U} - \boldsymbol{U}_m \cdots \boldsymbol{U}_1\| \leq \epsilon\ .$$

For example, such a universal gate set is given by $\{\mathbf{CX}, \mathbf{R_X}(\theta), \mathbf{R_Y}(\theta), \mathbf{R_Z}(\theta)\}$ but there are many other gate combinations. An exact representation for an arbitrary unitary $\boldsymbol{U} = \boldsymbol{U}_m \cdots \boldsymbol{U}_1$ is not possible since the number of unitary matrices is uncountable. Quantum computations are carried out with this universal ability of approximating unitaries, e.g., by computing the unitary corresponding to a *discrete fourier transformation* [98] efficiently. Keep in mind that these unitaries are of size $2^n \times 2^n$, making such computations intractable for classical computers.

Example 1 (Quantum Kernel). For showing an exemplary quantum circuit and bridging the gap to ML, we examine *quantum kernels*. Instead of relying on problem specific knowledge to construct the feature map or the kernel function, the intrinsically 2^d-dimensional Hilbert space of an d-qubit register is utilized to realize the feature map [99]. The feature space transformation is created by encoding the

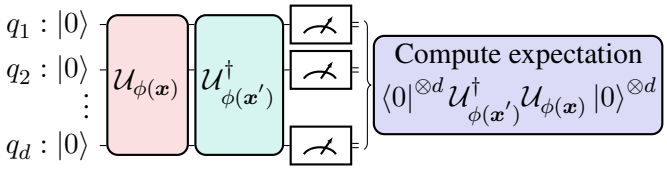


Figure 2.4: Circuit for computing the quantum kernel matrix. Given data points $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$, the d-qubit basis state $|0\rangle^{\otimes d}$ is prepared. First $\boldsymbol{x}$ is encoded into the circuit by applying $\mathcal{U}_{\phi(\boldsymbol{x})}$, leading to a feature representation in the quantum state. Similarly, $\boldsymbol{x}'$ is encoded by using $\mathcal{U}^{\dagger}_{\phi(\boldsymbol{x})}$ and through measurements of the outcome with respect to the projection operator $|0\rangle^{\otimes d} \langle 0|^{\otimes d}$, we can estimate the kernel value $K(\boldsymbol{x}, \boldsymbol{x}')$.

data into the underlying quantum circuit.

Assume we are given data points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$. The data does not enter the circuit in the discrete qubit state space but is passed to the circuit in form of parameters of universal unitary gates $\boldsymbol{U}_{\phi(\boldsymbol{x})}$ representing (parts of) the high-dimensional feature map with $\phi : \mathbb{R}^d \to \mathbb{R}^{2^d}$. Each classical d-bit binary string is interpreted as one feature and the corresponding probability amplitudes of the qubit state as feature values. The actual kernel value $K(\boldsymbol{x}, \boldsymbol{x}')$ is then given by estimating the transition amplitude

$$K(\boldsymbol{x}, \boldsymbol{x}') = \left|\langle \psi(\boldsymbol{x}) | \psi(\boldsymbol{x}') \rangle\right|^2 = \left|\langle 0|^{\otimes d} \mathcal{U}^{\dagger}_{\phi(\boldsymbol{x}')} \mathcal{U}_{\phi(\boldsymbol{x})} |0\rangle^{\otimes d}\right|^2 , \tag{2.21}$$

with $|\psi(\boldsymbol{x})\rangle = \mathcal{U}_{\phi(\boldsymbol{x})} |0\rangle^{\otimes d}$ and $\mathcal{U}_{\phi(\cdot)} = (\boldsymbol{U}_{\phi(\cdot)})^l$. An circuit diagram for computing this kernel value is given in Figure 2.4. Even though $\boldsymbol{U}_{\phi(\boldsymbol{x})}$ is universal for choosing $\boldsymbol{x}$ appropriately, the underlying data is not arbitrary. Thus, for the final feature map $\mathcal{U}_{\phi(\boldsymbol{x})}$, the application of $\boldsymbol{U}_{\phi(\boldsymbol{x})}$ is repeated for l times. This is known to increase the expressiveness of the overall circuit and is denoted as *data reuploading* [100]. Clearly, the specific choice of $\boldsymbol{U}_{\phi(\cdot)}$—known as *ansatz*—is not fixed and can be tuned for the application at hand. Finding the right ansatz choice is a major challenge in QML, and is largely dependent on the underlying data and quantum hardware specifics. In [99], the authors suggest the following unitary:

$$\boldsymbol{U}_{\phi(\boldsymbol{x})} = \exp\left(-i \sum_{\mathcal{S} \subset [d]} \phi_{\mathcal{S}}(\boldsymbol{x}) \prod_{k \in \mathcal{S}} \mathbf{Z}_k\right) \mathbf{H}^{\otimes d} , \tag{2.22}$$

where $\mathbf{Z}_k := \boldsymbol{I}_2^{\otimes k-1} \otimes \mathbf{Z} \otimes \boldsymbol{I}_2^{\otimes d-k}$ represents the Pauli-Z operator applied to qubit k.

Obtaining the full kernel matrix for n data points requires $n(n+1)/2$ evaluations of the circuit in Equation (2.21), due to the symmetry of the resulting kernel matrix. "Circuit evaluation" means, that we run the circuit multiple times, measure the corresponding output (usually a few thousand times) and approximate the expectation value by averaging.

For compatibility with current quantum devices, considering local feature functions for all subsets $\mathcal{S} \subset [d]$ is too costly. More details on limitations of current quantum hardware will follow in Section 2.4.4 and a hardware-aware formulation of $\boldsymbol{U}_{\phi(\boldsymbol{x})}$ is discussed later in Chapter 5.

2.4.2 Adiabatic Quantum Computing

AQC [33] operates on a fundamentally different principle compared to QGC. In QGC, computations can evolve throughout the entire Hilbert space and are implemented using a sequence of unitary quantum

logic gates. In contrast, AQC begins with an initial Hamiltonian whose ground state is easy to prepare and gradually transforms into a final Hamiltonian, whose ground state represents the solution to the computational problem. In quantum mechanics, a Hamiltonian is a mathematical operator that represents the total energy of a quantum system, including both kinetic and potential energy. It plays a central role in determining the time evolution and behaviour of quantum states. The *adiabatic theorem* [101, 102] ensures that, as long as the Hamiltonian changes slowly enough, the system remains in its instantaneous ground state (state with minimum energy) throughout the process.

AQC began as a method for solving optimization problems [102] and has since developed into a significant and universal alternative to QGC. Specific algorithms have been developed for AQC such as the one in [103] but it was shown that both QC paradigms are equivalent [104].

Definition 2.17 (Hamiltonian)**.** A Hamiltonian $\boldsymbol{H} \in \mathbb{C}^{2^n \times 2^n}$ is a Hermitian matrix and describes the energy of a quantum system with n qubits. If we conduct a measurement with respect to that Hamiltonian, we can end up in 2^n different energy configurations, which coincide with the eigenvalues $E^1_{\boldsymbol{H}} < E^2_{\boldsymbol{H}} < \dots$ of $\boldsymbol{H}$. The eigenvector corresponding to eigenvalue $E^1_{\boldsymbol{H}}$ is called ground state and the one corresponding to $E^1_{\boldsymbol{H}}$ is called first excited state.

Definition 2.18 (Schrödinger)**.** An evolution of a quantum state $|\psi\rangle$ under a time-dependent Hamiltonian $\boldsymbol{H}(t)$ is governed by the *Schrödinger* equation

$$i\hbar \frac{\partial |\psi(t)\rangle}{\partial t} = \boldsymbol{H}(t) |\psi(t)\rangle \,, \; |\psi(0)\rangle \equiv |\psi\rangle \;\; , \tag{2.23}$$

where $\hbar$ is the reduced *Planck* constant. Instead of solving the differential equation in Equation (2.23) with intractable classical methods, QC hardware can naturally perform this evolution.

In AQC we are given an initial Hamiltonian $\boldsymbol{H}_\mathrm{I}$ with known ground state $|\psi_{\boldsymbol{H}_\mathrm{I}}\rangle$ and want to find the ground state of target Hamiltonian $\boldsymbol{H}_\mathrm{P}$. This is done by evolving $|\psi_{\boldsymbol{H}_\mathrm{I}}\rangle$ sufficiently slow (adiabatically) under the Hamiltonian $\boldsymbol{H}(s) = A(s)\boldsymbol{H}_\mathrm{I} + B(s)\boldsymbol{H}_\mathrm{P}$ to end up in a ground state $\psi_{\boldsymbol{H}_\mathrm{P}}$ of $\boldsymbol{H}_\mathrm{P}$. $A, B : [0,1] \to [0,1]$ are monotonically decreasing and increasing, respectively, and $A(0) = B(1) = 1$, $A(1) = B(0) = 0$, i. e., $\boldsymbol{H}(0) = \boldsymbol{H}_\mathrm{I}$ and $\boldsymbol{H}(1) = \boldsymbol{H}_\mathrm{P}$. Setting $A(s) = 1 - s$ and $B(s) = s$ is a common choice. We assume $s \equiv t/T \in [0,1]$ to be a dimensionless time, such that $\boldsymbol{H}(s)$ is independent of T, where t is the actual time in the evolution and T is the total evolution time. The question remains how "long" we let our system evolve. This is governed by the adiabatic theorem, which tells us at which rate the Hamiltonian can be changed without leaving the ground state. With sufficient smooth variation of $\boldsymbol{H}(t)$, the time T scales as

$$T \in \mathcal{O}\left(\max_{s\in[0,1]} \frac{1}{\gamma^2_{\boldsymbol{H}(s)}} \right) , \quad \gamma_{\boldsymbol{H}(s)} \coloneqq E^2_{\boldsymbol{H}(s)} - E^1_{\boldsymbol{H}(s)} \;, \tag{2.24}$$

where $\gamma_{\boldsymbol{H}(s)}$ is called the optimum SG of $\boldsymbol{H}(s)$ or just SG in short.

Ising models representing certain NP-hard problems have been shown to exhibit exponentially small SG [105]. Often, we do not have much knowledge about the SG of the Hamiltonian at hand. In fact, determining or making assumptions about the SG is actually notoriously challenging [106]. How the structure and underlying properties of the given problem influences the SG will be investigated in Chapter 3. A vital assumption for this process is that the initial Hamiltonian $\boldsymbol{H}_\mathrm{I}$ has a unique ground state. Furthermore, it has to be ensured that the Hamiltonians do not commute, i. e., $\boldsymbol{H}_\mathrm{I}\boldsymbol{H}_\mathrm{P} \neq$

$\boldsymbol{H}_{\mathrm{P}}\boldsymbol{H}_{\mathrm{I}}$. Since $\boldsymbol{H}_{\mathrm{P}}$ is diagonal in the computational basis, this leads to transversity and introduces quantum entanglement. Through quantum fluctuations, the barriers of the energy landscape can be traversed efficiently through *quantum tunneling* effects. If the Hamiltonians commute, their share a a common *eigenbasis*. That is, they share the eigenstates and just differ on the corresponding energy values/eigenvalues. During evolution, the energy levels evolve trivially as weighted sums of the corresponding eigenvalues, i. e., the system remains in the initial ground state with transition to other states. If the ground state of $\boldsymbol{H}_{\mathrm{I}}$ coincides with the one of $\boldsymbol{H}_{\mathrm{P}}$, we would end up in a ground state of $\boldsymbol{H}_{\mathrm{P}}$ after the evolution. However, the assumption is that finding a ground state of $\boldsymbol{H}_{\mathrm{P}}$ is hard, so it would be infeasible to prepare a ground state for $\boldsymbol{H}_{\mathrm{I}}$. In the opposite case, an energy level crossing would exist, which leads the SG to vanish during the evolution process, letting the adiabatic theorem fail.

2.4.3 Quantum Optimization

QO refers to the use of QC techniques and principles to solve optimization problems. As we have seen earlier, problems include areas like logistics, finance, ML, drug discovery, and energy management. QO can offer significant speed advantages over classical approaches, especially as the size and complexity of the problem grow. Quantum devices may explore large solution spaces more efficiently due to their ability to operate in parallel over quantum states.

The original idea for QO comes from encoding CO problems into quantum Hamiltonians [102]. It uses the observation that the Ising model (Definition 2.8) can be conveniently translated to the quantum realm by using the Ising Hamiltonian of an n-qubit system

$$\boldsymbol{H}_{\boldsymbol{J},\boldsymbol{h}} = \sum_{i,j=1}^{n} J_{ij}\mathbf{Z}_i\mathbf{Z}_j + \sum_{i=1}^{n} h_i\mathbf{Z}_i \ . \tag{2.25}$$

Since $\mathbf{Z}$ is diagonal, $\mathbf{Z}_i$ is diagonal $\forall i \in [n]$ and thus $\boldsymbol{H}_{\boldsymbol{J},\boldsymbol{h}}$ is also diagonal. It holds that $\mathrm{diag}(\mathbf{Z}) = (1,-1)^\top$ and due to the nature of the Kronecker product, the diagonal of $\boldsymbol{H}_{\boldsymbol{J},\boldsymbol{h}}$ exactly consists of all possible Ising energy configurations in lexicographical order. Hence, finding the ground state energy (minimum eigenvalue) of $\boldsymbol{H}_{\boldsymbol{J},\boldsymbol{h}}$ is equivalent to finding a minimum of the Ising model

$$\min_{|\psi\rangle\in\mathbb{C}^{2^n}} \langle\psi|\, \boldsymbol{H}_{\boldsymbol{J},\boldsymbol{h}}\, |\psi\rangle = \min_{\boldsymbol{\sigma}\in\mathbb{S}^n} \boldsymbol{\sigma}^\top \boldsymbol{J}\boldsymbol{\sigma} + \boldsymbol{h}^\top\boldsymbol{\sigma} \ .$$

Furthermore, due to diagonality, every eigenstate (eigenvector) $|\psi\rangle$ of $\boldsymbol{H}_{\boldsymbol{J},\boldsymbol{h}}$ can be factorized by the single-qubit basis states, that is $|\psi\rangle = |\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle$, $|\psi_i\rangle \in \{|0\rangle, |1\rangle\}\ \forall i \in [n]$. Thus, with finding a ground state $|\psi\rangle$ of $\boldsymbol{H}_{\boldsymbol{J},\boldsymbol{h}}$ we also obtain a minimizer $\boldsymbol{\sigma}$ to the corresponding Ising model. As we have seen in Proposition 2.1, every QUBO problem can be converted to an Ising problem and vice versa. This makes QC a prominent candidate for solving CO problems.

We examine two different techniques of finding a ground state the of Ising Hamiltonian: one is based on QGC and the other one on AQC.

Variational Quantum Algorithms A Variational Quantum Algorithm (VQA) is targeted to optimization problems for near-term quantum devices which uses the *variational principle* in quantum theory. The variational principle is a method for determining the lowest expectation value of a given observable, typically the ground state energy of a Hamiltonian $\boldsymbol{H}$, by evaluating it with respect to a trial

state $|\psi(\boldsymbol{\theta})\rangle$. This principle is termed "variational" because the trial state is parameterized by a set of adjustable parameters $\boldsymbol{\theta}$, enabling a general form of the state to be optimized for the system, thereby identifying the minimum expectation value.

We aim to optimize $\langle\psi(\boldsymbol{\theta})|\,\boldsymbol{H}\,|\psi(\boldsymbol{\theta})\rangle$ using a hybrid quantum-classical approach. For this purpose, the trial state can be written as $|\psi(\boldsymbol{\theta})\rangle = \boldsymbol{U}(\boldsymbol{\theta})\,|0\rangle^{\otimes n}$, where $\boldsymbol{U}(\boldsymbol{\theta})$ is a unitary matrix implementable as a quantum circuit on QGC hardware and is called *ansatz*. The circuit parameters $\boldsymbol{\theta}$ are optimized using a classical optimization routine to estimate the ground state. In the literature this is refered to as a Variational Quantum Eigensolver (VQE) [107].

The choice of ansatz form is a distinct challenge with numerous approaches that vary depending on the system's Hamiltonian [108], and is often guided by the specific problem context. For instance, in quantum chemistry, estimating the ground state of a helium atom might begin with an ansatz formed as the product of two hydrogen atom wave functions [109]. In addition to a problem-inspired ansatz, another approach is a hardware-efficient ansatz [110], which uses the gate set native to the available quantum hardware. Its key advantages include reducing circuit depth and offering versatility, as it is "problem-agnostic" and can be applied to a broad range of problems. The general pipeline for a VQA is depicted in Figure 2.5 for a specific ansatz tailored towards solving CO problems.

Example 2 (QAOA). The Quantum Approximate Optimization Algorithm (QAOA) [111] is a hybrid quantum-classical VQA designed for solving QUBO problems. It employs a specific quantum circuit ansatz with alternating layers of a target Hamiltonian and a mixing Hamiltonian and approximates the solution to a problem encoded in a classical cost function. QAOA has later been extended to the Quantum Alternate Operating Ansatz [112] by allowing a broader class of operators to handle constraints directly, or operate in spaces beyond binary variables. In this thesis, we mainly deal with QUBO problems and thus we focus on the original definition of QAOA.

The idea behind QAOA is the approximation of the time evolution in AQC using QGC, which is called *trotterization*. It is based on *Trotter's product formula* [113] for complex matrices

$$\exp\left(\boldsymbol{A}+\boldsymbol{B}\right) = \lim_{l\to\infty}\left(\exp\left(\boldsymbol{A}/l\right)\exp\left(\boldsymbol{B}/l\right)\right)^{l},\ \boldsymbol{A},\boldsymbol{B}\in\mathbb{C}^{N\times N},$$

which is also known as the *first-order Suzuki-Trotter expansion*. Note that the time-dependent evolution operator $\boldsymbol{U}(s)$ of the AQC Hamiltonian $\boldsymbol{H}(s) = A(s)\boldsymbol{H}_{\mathrm{I}} + B(s)\boldsymbol{H}_{\mathrm{P}}$ can be written as

$$\begin{aligned}
\boldsymbol{U}(s) &= \exp\left(-i\int_0^s \boldsymbol{H}(\tau)\,d\tau\right) && \text{(2.26a)}\\
&= \exp\left(-i\int_0^s \left(A(\tau)\boldsymbol{H}_{\mathrm{I}} + B(\tau)\boldsymbol{H}_{\mathrm{P}}\right)\,d\tau\right) && \text{(2.26b)}\\
&= \exp\left(-i\int_0^s A(\tau)\,d\tau\boldsymbol{H}_{\mathrm{I}} - i\int_0^s B(\tau)\,d\tau\boldsymbol{H}_{\mathrm{P}}\right) && \text{(2.26c)}\\
&\approx \prod_{k=1}^{l}\exp\left(-i\int_0^s \frac{A(\tau)}{l}\,d\tau\boldsymbol{H}_{\mathrm{I}}\right)\exp\left(-i\int_0^s \frac{B(\tau)}{l}\,d\tau\boldsymbol{H}_{\mathrm{P}}\right), && \text{(2.26d)}
\end{aligned}$$

where Equation (2.26a) is the general solution to the Schrödinger equation Equation (2.23), i. e., $|\psi(s)\rangle = \boldsymbol{U}(s)\,|\psi\rangle$ and Trotter's product formula is applied in Equation (2.26d). Interestingly, the matrix $\exp(-i\boldsymbol{H})$ is unitary if $\boldsymbol{H}$ is Hermitian and thus can be implemented by a sequence of quantum gates. Thus, the continuous evolution given by Equation (2.26a) can be emulated with QGC in sufficiently

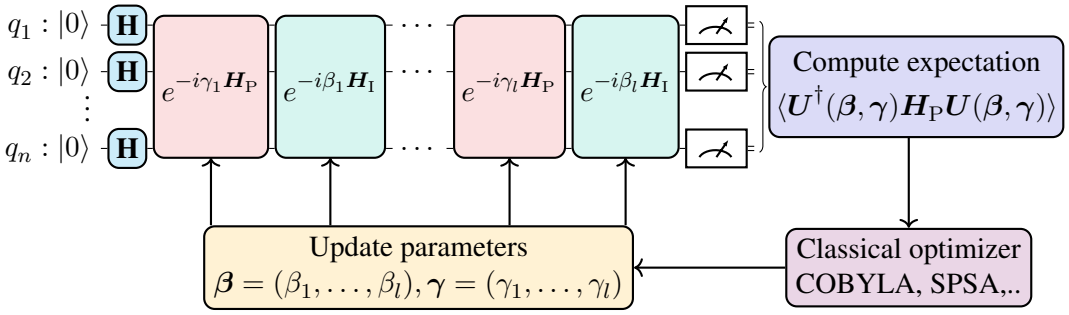


Figure 2.5: VQA pipeline overview: A parameterized quantum circuit is optimized w.r.t. the expectation of an observable in a hybrid quantum-classical loop. As an example, we depict the QAOA circuit along with the objective of finding a minimum eigenstate of the problem Hamiltonian $\boldsymbol{H}_\mathrm{P}$.

small steps.

To enhance the approximation in Equation (2.26d), QAOA implements the circuit

$$\boldsymbol{U}(\boldsymbol{\beta},\boldsymbol{\gamma}) = \prod_{k=1}^{l} \exp\left(-i\beta_k \boldsymbol{H}_\mathrm{I}\right) \exp\left(-i\gamma_k \boldsymbol{H}_\mathrm{P}\right), \ \boldsymbol{\beta},\boldsymbol{\gamma} \in \mathbb{R}^l \,, \tag{2.27}$$

where $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are optimizable parameter vectors. This parameterized circuit is depicted in Figure 2.5. QAOA is initalized by (i) constructing $\boldsymbol{H}_\mathrm{P}$ which encodes the given QUBO problem using Equation (2.25), (ii) defining $\boldsymbol{H}_\mathrm{I}$, (iii) choosing the number of layers $l \geq 1$ and constructing the circuit $\boldsymbol{U}(\boldsymbol{\beta},\boldsymbol{\gamma})$ in Equation (2.27) and (iv) preparing an initial state $|\psi_0\rangle$. After initialization, $\langle\psi_0| \boldsymbol{U}^\dagger(\boldsymbol{\beta},\boldsymbol{\gamma}) \boldsymbol{H}_\mathrm{P} \boldsymbol{U}(\boldsymbol{\beta},\boldsymbol{\gamma}) |\psi_0\rangle$ is evaluated with the help of measurements and the parameters $\boldsymbol{\beta},\boldsymbol{\gamma}$ are optimized with a classical computer. Due to the probabilistic nature of QC, it is often hard to compute a gradient analytically and thus often falls back to derivative-free optimizers, such as COBYLA [114] or SPSA [115].

$\boldsymbol{H}_\mathrm{I}$ is called the mixer Hamiltonian and is often chosen as $\boldsymbol{H}_\mathrm{I} = -\sum_{i=1}^{n} \mathbf{X}_i$, where $\mathbf{X}_i := \boldsymbol{I}_2^{\otimes i-1} \otimes \mathbf{X} \otimes \boldsymbol{I}_2^{\otimes n-i}$ represents the Pauli-X operator applied to qubit i. This choice might look arbitrary but it has certain key properties. In order for QAOA to avoid trivial dynamics, it is crucial that $\boldsymbol{H}_\mathrm{I}$ and $\boldsymbol{H}_\mathrm{P}$ do not commute ($\boldsymbol{H}_\mathrm{I}\boldsymbol{H}_\mathrm{P} \neq \boldsymbol{H}_\mathrm{P}\boldsymbol{H}_\mathrm{I}$), which is the case, since $\mathbf{XZ} \neq \mathbf{ZX}$. This ensures that the optimization process involves genuinely quantum phenomena, such as entanglement and state exploration. Furthermore, the initial state $|\psi_0\rangle$ should correspond to the ground state of $\boldsymbol{H}_\mathrm{I}$ and should be easy to propose. This is the case since the equal superposition state $|0\rangle^{\otimes n}$ is the ground state. Lastly, $\boldsymbol{H}_\mathrm{I}$ is rather easy to implement in current quantum hardware.

Quantum Annealing The concept of encoding the solution to a computational problem within the ground state of a quantum Hamiltonian first emerged in 1988 in the context of classical CO, where it was referred to as *quantum stochastic optimization* or QA [116–118]. These early works highlighted that QA should be viewed as an algorithm leveraging simulated quantum fluctuations and tunneling, rather than thermal fluctuations, thereby offering a quantum-inspired counterpart to SA.

Nowadays, QA denotes a physical hardware implementation of AQC [28, 119–122] instead of emulating it with the aforementioned methods. QA devices offer an alternative approach to QGC for achieving quantum-enhanced information processing, focusing on similar application areas such as

CO and generative learning. It cannot only be used for optimization but also for efficient sampling from complex probability distributions. This is very crucial in probabilistic graphical models such as *Bayesian Networks* and *Markolv Random Fields* [71] but also for training the powerful ANN model of *Boltzmann machines* [123, 124]. However, one should keep in mind that QA is an imperfect physical implementation and thus a heuristic for AQC.

2.4.4 Limitations of Quantum Hardware

Over the past decade, quantum computers have been become more and more accessible. One of the state-of-the-art QGC hardware providers is IBM, which launched a cloud-based service in 2016, allowing public members to access their quantum devices[3]. As of 2025[4], 11 systems—each with at least 127 physical qubits—are available for running custom quantum circuits. The interface between circuit design and execution is given by the Python package *Qiskit*. It is designed for compiling quantum circuits and executing them on quantum devices (*backends*). Once executed, the measurement results are sent back to the user. The execution of circuits is subject to several constraints, including limits on the number of circuits, the number of measurements, the circuit depth and sharing a queue with other users. When backend workloads are high, experiments may experience significant delays due to extended queue times.

A similar cloud-access is given by the company D-Wave with their *Leap*[5] platform. Contrary to IBM, D-Wave does not (yet) offer access to a universal quantum gate computer but to quantum annealers with up to 5670 qubits. These systems are specifically tailored towards optimizing QUBO problems or the equivalent Ising models, respectively. For fine-tuning the behaviour during the annealing process, many different parameters can be set manually, such as the annealing time T.

Even though the public accessability sounds promising for achieving a near-term quantum advantage, there are several limitations of the current generation of quantum computers to be mentioned. As of now, we find ourselves in the NISQ era [29]. This term describes the emergence of quantum devices with hundreds to thousands of qubits, which are, however, largely susceptible to errors. Full-scale quantum error correction is not yet practical, as it requires a significant number of physical qubits to encode a single logical qubit [125]. Consequently, NISQ devices function in an intermediate regime, bridging the gap between classical computation and fully fault-tolerant quantum computers.

Noise A major error source for NISQ devices is the *decoherence* of the quantum state which leads to the loss of superposition and entanglement [126]. Quantum systems must be isolated to maintain their fragile quantum states, but real-world devices suffer from different unwanted physical side-effects, such as thermal noise, electromagnetic radiation or cosmic rays. In quantum devices, this effect is measured via the decoherence time, i.e. how long a qubit can maintain its quantum state. Furthermore, quantum gate operations are imperfect due to control inaccuracies, hardware imperfection, and crosstalk, describing unintended interactions between nearby qubits. Even though error mitigation and suppression techniques such as *zero-noise extrapolation* [127] or *dynamic decoupling* [128] are being investigated, the aforementioned caveats limit the depth and complexity of executable quantum circuits.

[3] https://www.ibm.com/quantum/blog/quantum-five-years (last accessed September 19, 2025)
[4] https://quantum.ibm.com/services/resources (last accessed September 19, 2025)
[5] https://cloud.dwavesys.com/leap (last accessed September 19, 2025)

Another limiting factor is the underlying hardware topology, i. e., the physical layout of the qubits describing how they are interconnected. For example, IBM uses the *heavy hex-lattice* topology[6], featuring qubits in a hexagonal pattern, where each qubit is linked to two or three neighbors. This connectivity limits the application of arbitrary multi-qubit gates, which have to be broken down into a gate sequence for neighboring qubits, denoted as *transpilation*. Most circuits must undergo a series of transformations that make them compatible with a given target device, and optimize them to reduce the effects of noise of the resulting outcomes. Rewriting quantum circuits to match hardware constraints and optimizing for performance can be far from trivial. The flow of logic in the rewriting tool chain need not be linear, and can often have iterative subloops, conditional branches, and other complex behaviors. Most importantly, it encompasses the decomposition of gates involving three or more qubits into hardware-native two-qubit gates. As a direct result, an apparently "shallow" quantum gate circuit, consisting of a single unitary operation among n-qubits, can thus eventually exhibit a very high depth.

Even though quantum annealers do not rely on the design of sophisticated quantum circuits, they also suffer from different error sources, such as *integrated control errors*. A more detailed explanation is given in Chapter 4, where we address the mitigation of these effects.

Intermediate Scale The number of physical qubits in current quantum systems ranges up to a few thousand. This can already be leveraged for simulating small-scale quantum-mechanical systems [129]. However, they are not yet fit for large-scale applications appearing in many CO and ML problems [130]. Furthermore, the sheer number of physical qubits is not directly decisive for the devices' ability to solve problems of that size.

Even though the qubit size of 5670 might sound impressive for the D-Wave Advantage System[7], the qubit connectivity is restricted. For embedding arbitrary graphical structures into the underlying topology, D-Wave uses the method of chaining multiple physical qubits together to a single super-qubit, representing a single logical qubit of the problem at hand. To embed arbitrary QUBO problems, the current *Pegasus* topology[8] allows for up to 182 densely connected qubits, while the upcoming *Zephyr*[9] topology for 232. Combined with the noise limitations mentioned before, the full quantum chip (all phyiscal qubits) of NISQ hardware cannot be completely utilized, leading to a current applicability of problems to a, at most, a few hundred qubits.

Compared to digital computing, QC technology remains immature due to fundamental challenges in hardware scalability, error correction, and algorithmic development. Unlike classical transistors, which have undergone decades of miniaturization and optimization under *Moore's Law*, qubits are highly sensitive to noise and decoherence, limiting their reliability and requiring complex error correction schemes that significantly increase qubit overhead [29]. Additionally, while classical algorithms have been refined for decades across various architectures, quantum algorithms remain niche, with limited practical advantages over classical methods for most real-world problems.

On the other hand, QC hardware strongly varies in qubit scalability, runtime efficiency, and susceptibility to errors. Superconducting qubits, used by IBM, Google and D-Wave, offer fast gate speeds (nanoseconds) and moderate scalability ($\sim$ 100s of qubits) but suffer from short coherence times

[6] https://www.ibm.com/quantum/blog/heavy-hex-lattice (last accessed September 19, 2025)

[7] https://www.dwavesys.com/solutions-and-products/systems (last accessed September 19, 2025)

[8] https://www.dwavesys.com/media/jwwj5z3z/14-1026a-c_next-generation-topology-of-dw-quantum-processors.pdf (last accessed September 19, 2025)

[9] https://www.dwavesys.com/media/2uznec4s/14-1056a-a_zephyr_topology_of_d-wave_quantum_processors.pdf (last accessed September 19, 2025)

($\sim 100\ \mu s$) and high gate errors ($\sim 0.1\%$) [131]. Trapped ion qubits, such as those developed by IonQ, provide longer coherence times (seconds) and high-fidelity gates ($\sim 99.9\%$) but have slower operations (ms per gate) and challenges in scaling beyond 100 qubits [132]. Neutral atom qubits, pioneered by QuEra, promise better scalability (~ 1000 qubits) and long coherence, though gate fidelities are still improving [122]. Photonic quantum computers, like those from Xanadu, leverage linear optics for high-speed operations and room-temperature stability but require complex error correction due to photon loss and are not universal [133]. Microsoft builds upon topological qubits, which are small and fast, but the scalability claim has yet to be proven [134]. The lack of standardized quantum hardware, varying qubit modalities (superconducting, trapped ions, etc.), and the need for ultra-low-temperature or vacuum environments further hinder commercialization.

Until scalable, error-corrected quantum systems emerge, classical computers will continue to dominate practical computing tasks. Despite its current immaturity, QC research is crucial because it has the potential to solve problems that are intractable for classical computers. Even in the near term, NISQ devices may provide advantages in specialized applications, such as in QML. In the first part of this thesis (Chapters 3 and 4), we thus investigate the effect of data properties of an underlying ML problem on the noise sensitivity in NISQ devices and how to mitigate them. Coping with the intermediate-scale component is investigated in the second part (Chapters 5 to 7), where we develop efficient algorithms for decomposing a large-scale problem into suitable NISQ-aware subproblems. We show their effectiveness on three different real-world problems, whose huge dimensionality stems from the underlying data.

Part I

Effects of Data Complexity on Quantum Optimization

CHAPTER 3

Relating Data Complexity to Solvability

QO leverages techniques based on AQC, such as QA and variational algorithms, to solve complex CO problems more efficiently than classical methods. In the NISQ era, current quantum devices contain hundreds to thousands of qubits but suffer from noise, limited coherence times, and gate errors, restricting their ability to outperform classical computers in practical applications. While NISQ devices might enable early demonstrations of quantum advantage in optimization tasks, their limitations—such as error rates, qubit connectivity constraints, and the lack of robust error correction—pose significant challenges.

In this first part of the thesis, we are interested in the effect of different data characteristics on the performance of QO methods and how to mitigate possible difficulties. In this chapter, we first aim to understand the relation between data complexity and the *quantum solvability*. That is, we examine the SG of the corresponding Hamiltonian of a given QUBO problem, which is directly related to the system evolution time in AQC. The effect of data complexity on QUBO formulations can be characterized by observing that CO problems lie at the core of many NP-hard problems in ML [135]: clustering a dataset comes down to deciding for every point, whether it belongs to one cluster or another. Training an SVM involves identifying the subset of SVs. These decisions are highly interdependent, making the tasks computationally complex. Data complexity properties can then be chosen as parameters that characterize the hardness of ML problems. For example, clustering can be very prone to outliers (compactness), while SVMs are largely dependent on the separability of the underlying data. An overview of the investigation of data complexity on the solvability with QO is given in Figure 3.1. Apart from considering the SG as a measure for solvability, which is a general factor in every AQC-based QO method, we examine limitations specific to NISQ devices in Chapter 4. In particular, we connect data complexity with the proneness to errors by describing an effective precision measure, and develop efficient techniques for mitigating the noise in NISQ devices.

Given some dataset, the question arises of how to obtain a quantum-compatible optimization format describing a certain ML task. In Section 3.2, we answer this question by presenting QUBO embeddings for the three different ML tasks discussed in Chapter 2: SVM, clustering and VQ. We show how constraints can be integrated into the objective function for obtaining equivalence to the underlying QBP formulation. Further, QUBO equivalence is proven between KMEDVQ and MDVQ, which leads to an interesting insight into kernel alignment and distance-based VQ methods. A more profound understanding of the SG of the problem Hamiltonian is provided in Section 3.3, along with its relation to the AQC Hamiltonian. Since the SG directly relates to the evolution time needed for AQC to be successful, it can be used as an appropriate metric for solvability. A large SG implies a shorter runtime

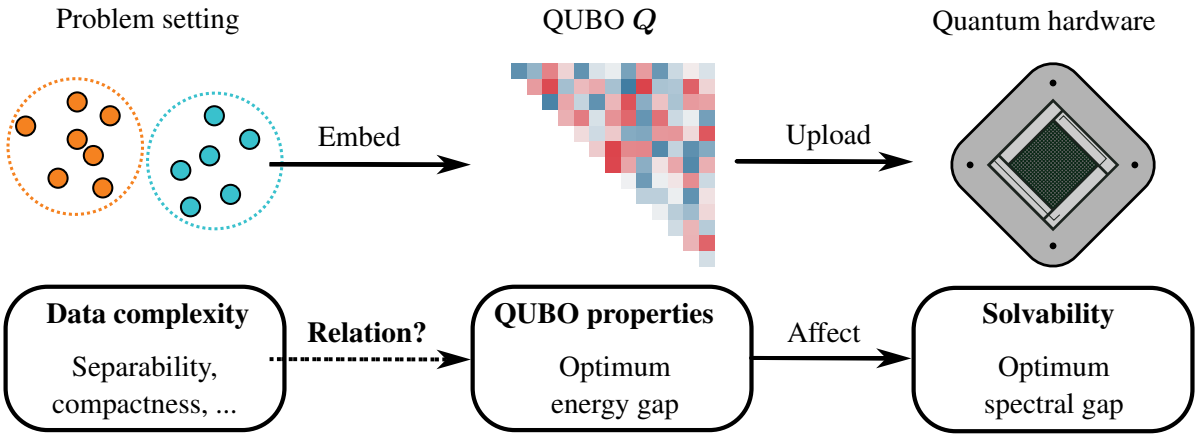


Figure 3.1: Schematic depiction of the effect of data complexity on QUBO solvability with QO. We are given some CO problem—in this case biclustering—which is embedded into a QUBO formulation. Properties describing the complexity of the underlying data (e. g. separability or compactness) affect the properties of the corresponding QUBO matrix, such as the optimum energy gap (see Equation (3.11)). These properties then affect the performance of quantum hardware in solving the QUBO, in terms of the optimum SG of the corresponding quantum Hamiltonian. Thus, we investigate the relation between data complexity and corresponding QUBO properties.

for obtaining an optimal solution, while a small SG can lead to the problem being unsolvable in a reasonable amount of time.

The SG is a physical property of the AQC Hamiltonian, and its connection to classical complexity theory is poorly understood. One would expect that classically hard problems tend to be more difficult to solve, even when non-classical QO methods are used. We thus investigate the relation between data complexity and the SG of the corresponding QUBO formulation. An experimental evaluation for SVM and clustering is conducted in Section 3.4, where we vary properties such as separability, compactness, and cluster size. Surprisingly, we find that the relationship does not always align with intuition: with clustering, a stronger separation of data corresponds with a larger SG, while for SVM learning the opposite is true. Moreover, the incorporation of constraints can also have a negative effect on the SG without a reasonable choice for the penalty parameters.

3.1 Related Work

Obtaining theoretical general estimates on the SG is notoriously hard [106]. Thus, methods have been developed for obtaining bounds in specific scenarios. For example, the authors of [136] obtain polynomial bounds dependent on the number of quantum gates used in simulating a quantum circuit with AQC. For assuming *first-order quantum phase transitions* during the adiabatic evolution, the use of perturbation expansion for computing a lower bound on the SG is proposed in [137]. This concept has been more refined in [138], by using graph-theoretical methods and degenerate perturbation theory for obtaining bounds on the location of the SG during the evolution.

In [139], the effect of changing qubit interactions on the SG is investigated for the *Maximum-weighted Independent Set* problem. Varying this interactions can lead to an increased SG and thus better performance for AQC. However, it has been shown, that this framework is not generally applicable to arbitrary QUBO formulations, as shown with the counter example of *Weighted Max k-Clique* in [140].

Altering the evolution path between the initial and the problem Hamiltonian has been shown to enlargen the SG [141]. Further, using higher-dimensional problem formulations instead of QUBO can have advantageous effects [142].

Expanding on these insights, the authors of [143] propose a technique to modify the penalty factors of constrained Hamiltonians, thereby improving their SGs. The subsequent work in [144] explored patterns and guidelines for setting the appropriate penalty factors to achieve an overall improvement in solution quality. Varying penalty parameters and reformulating QUBO problems for permutation-based optimization has been investigated in [145]. A more general approach is taken in [146] by estimating a penalty parameter using efficiently computable bounds.

Although all of the aforementioned methods aim to obtain insights on the SG for improving the performance of AQC, they are mainly tailored towards specific use-cases or settings. Moreover, most of these insights are either based on quantum effects during the adiabatic evolution process, such as anti-crossing or phase transitions, or manipulating penalty parameters in the given QUBO formulations. It has not yet been addressed how the underlying data affects the corresponding QUBO embedding, which will be investigated in this chapter.

3.2 QUBO Embeddings for Machine Learning

Before we can analyze the effect of data properties on QUBO solvability, we give different QUBO embeddings for well-known ML problems. In order to obtain such QUBO formulations, one needs a conversion from a constrained CO problem.

3.2.1 Integrating Linear Equality Constraints

Assume our problem can be formulated as a QBP (Definition 2.5) with linear integer equality constraints. We can obtain an equivalent QUBO formulation.

Proposition 3.1. *Let $\boldsymbol{W} \in \mathbb{R}^{n\times n}$, $\boldsymbol{A} \in \mathbb{Z}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{Z}^m$. There exists a vector $\boldsymbol{\lambda} \in \mathbb{R}^m_+$, such that the following equivalence holds*

$$\begin{aligned} &\min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{W}\boldsymbol{z} \\ &\text{s.t. } \boldsymbol{A}\boldsymbol{z} = \boldsymbol{b} \end{aligned} \quad \Leftrightarrow \quad \min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{W}\boldsymbol{z} + \sum_{i\in[m]} \lambda_i \left(\boldsymbol{A}_i^\top \boldsymbol{z} - b_i\right)^2 .$$

We can find an equivalent QUBO formulation with $\boldsymbol{\lambda}' := (\sqrt{\lambda_i})^\top_{i\in[m]}$, $\boldsymbol{A}' := \boldsymbol{\lambda}'\mathbf{1}^\top \odot \boldsymbol{A}$ denoting row-wise multiplication of $\boldsymbol{\lambda}'$ to $\boldsymbol{A}$ and $\boldsymbol{b}' := \boldsymbol{\lambda}' \odot \boldsymbol{b}$

$$\begin{aligned} &\min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{W}\boldsymbol{z} \\ &\text{s.t. } \boldsymbol{A}\boldsymbol{z} = \boldsymbol{b} \end{aligned} \quad \Leftrightarrow \quad \min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{Q}\boldsymbol{z}, \quad \boldsymbol{Q} := \boldsymbol{W} + \boldsymbol{A}'^\top \boldsymbol{A}' - 2\,\mathrm{diag}(\boldsymbol{b}'^\top \boldsymbol{A}') .$$

Proof. For showing necessity, assume that $\boldsymbol{z}^*$ is an optimizer of the LHS, i. e., $\boldsymbol{A}\boldsymbol{z}^* = \boldsymbol{b}$. Then

$$\boldsymbol{A}\boldsymbol{z}^* = \boldsymbol{b} \Leftrightarrow \boldsymbol{A}_i^\top \boldsymbol{z}^* = b_i \Leftrightarrow \left(\boldsymbol{A}_i^\top \boldsymbol{z}^* - b_i\right)^2 \ \forall i \in [m] .$$

Assume now that $\boldsymbol{A}\boldsymbol{z}^* \neq \boldsymbol{b}$, i.e., there exists $j \in [m]$, such that $\boldsymbol{A}_k^\top \boldsymbol{z}^* \neq b_j$. Due to the integer assumption, it follows that $\left(\boldsymbol{A}_j^\top \boldsymbol{z}^* - b_j\right)^2 > 1$. With choosing $\boldsymbol{\lambda} = \left(\sum_{i,j\in[n]} |W_{ij}|\right) \mathbf{1}$, we obtain

$$\sum_{i\in[m]} \lambda_i \left(\boldsymbol{A}_i^\top \boldsymbol{z}^* - b_i\right)^2 > \lambda_k = \sum_{i,j\in[n]} |W_{ij}| > \boldsymbol{z}^\top \boldsymbol{W} \boldsymbol{z} \quad \forall \boldsymbol{z} \in \mathbb{B}^n \, .$$

Thus, optimizing the RHS always leads to finding feasible solution adhering the constraints, inducing sufficiency. □

Corollary 3.1. *Let $\boldsymbol{W} \in \mathbb{R}^{n\times n}$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$. The following equivalence holds for large enough $\lambda > 0$*

$$\begin{aligned} &\min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{W} \boldsymbol{z} \\ &\text{s.t. } \boldsymbol{A}\boldsymbol{z} = \boldsymbol{b} \end{aligned} \quad \Leftrightarrow \quad \min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{W} \boldsymbol{z} + \lambda \left(\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}\right)^\top \left(\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}\right)$$

Finding a suitable value is far from trivial and is of great interest in current research [146]. The parameters λ_i have to be chosen large enough to ensure the constraints to be fulfilled. Choosing $\lambda_i = \sum_{i,j\in[n]} |W_{ij}|$ always guarantees this equivalence, which is however an optimistic upper bound. As we will see later on in Section 3.4, choosing these parameters too large has a negative effect on the solvability with QO. We go over to ML QUBO embeddings, in particular, we examine SVM, clustering, and VQ.

3.2.2 Binary Support Vector Machine

For simplicity, the definition of the dual SVM from Definition 2.10 is recollected for labeled data:

$$\max_{\boldsymbol{\alpha}} \mathbf{1}^\top \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^\top \left(\boldsymbol{y}\boldsymbol{y}^\top \odot \boldsymbol{K}\right) \boldsymbol{\alpha} \tag{3.1a}$$

$$\text{s.t. } \mathbf{0} \preceq \boldsymbol{\alpha} \preceq C\mathbf{1}, \tag{3.1b}$$

$$\boldsymbol{y}^\top \boldsymbol{\alpha} = 0 \, . \tag{3.1c}$$

C controls how "soft" the margin is, i.e., how strongly misclassified data points are penalized. A large C does so heavily, which may result in overfitting. A QUBO formulation is given through the following proposition.

Proposition 3.2 (BINSVM)**.** *Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i\in[n]} \subset \mathbb{R}^d \times \mathbb{S}$, $\boldsymbol{y} = (y_1, \ldots, y_n)^\top$ and $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a Mercer kernel with corresponding kernel matrix $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ respective the given dataset. A QUBO problem for the dual SVM is given by*

$$\min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{Q} \boldsymbol{z}, \quad \boldsymbol{Q} = \boldsymbol{y}\boldsymbol{y}^\top \odot \left(\frac{C}{2}\boldsymbol{K} + \lambda \mathbf{1}\mathbf{1}^\top\right) - \boldsymbol{I} \, . \tag{3.2}$$

Proof. Following [71] we make the simplifying assumption that α_i can only take the values 0 or C,

which allows us to introduce binary variables $z_i \in \mathbb{B}$ and write $\alpha_i = Cz_i$, leading to a QBP

$$\begin{aligned} \max_{\boldsymbol{z}\in\mathbb{B}^n} \; & C\mathbf{1}^\top \boldsymbol{z} - \frac{C^2}{2}\boldsymbol{z}^\top \left(\boldsymbol{y}\boldsymbol{y}^\top \odot \boldsymbol{K}\right)\boldsymbol{z} \\ & C\boldsymbol{y}^\top \boldsymbol{z} = 0 \; . \end{aligned}$$

The condition Equation (3.1c) can be included in the main objective by introducing the penalty term $-\lambda(\boldsymbol{y}^\top \boldsymbol{\alpha})^2$, which is zero when the condition is fulfilled, and negative otherwise. The parameter λ has to be chosen large enough, s.t., the constraint is fulfilled. From Equation (3.1), we obtain

$$\begin{aligned} & \max_{\boldsymbol{z}\in\mathbb{B}^n} \quad C\mathbf{1}^\top \boldsymbol{z} - \frac{C^2}{2}\boldsymbol{z}^\top \left(\boldsymbol{y}\boldsymbol{y}^\top \odot \boldsymbol{K}\right)\boldsymbol{z} - \lambda C \boldsymbol{z}^\top \boldsymbol{y}\boldsymbol{y}^\top \boldsymbol{z} \\ \Leftrightarrow \; & \min_{\boldsymbol{z}\in\mathbb{B}^n} \quad \boldsymbol{z}^\top \left(-\boldsymbol{I} + \frac{C}{2}\left(\boldsymbol{y}\boldsymbol{y}^\top \odot \boldsymbol{K}\right) + \lambda \boldsymbol{y}\boldsymbol{y}^\top\right)\boldsymbol{z} \; , \end{aligned}$$

concluding the proof. □

The assumption $\alpha_i = Cz_i$ is highly restrictive, as it permits no intermediate states between being a non-support vector ($\alpha_i = 0$) and being misclassified ($\alpha_i = C$). An extension with more precision is given in [147, 148], where the problems size is increased by a factor of $k > 0$, for approximating the interval $[0, C]$ in binary representation with k bits.

3.2.3 Biclustering

We recall the definition of the k-means objective given in Definition 2.12 for the simplest case of setting $k = 2$, also denoted as *biclustering*. However, it was shown that even in that case, it is NP-hard to find an optimal solution [149]. Assume we are given a set of n data points $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$. We want to partition $\mathcal{D}$ into disjoint clusters $C_1, C_2 \subset \mathcal{D}$, $C_1 \dot\cup C_2 = \mathcal{D}$. A strategy is to minimize the *within cluster scatter*,

$$\min_{C_1,\ldots,C_k \subset \mathcal{D}} \sum_{i\in[k]} \sum_{\boldsymbol{x}\in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2 \; , \quad \boldsymbol{\mu}_i \coloneqq \frac{1}{|C_i|}\sum_{\boldsymbol{x}\in C_i} \boldsymbol{x} \; , \tag{3.3}$$

where $\boldsymbol{\mu}_i$ is the mean of cluster C_i.

Proposition 3.3 (BICLUS [59])**.** *Let $n \in \mathbb{N}$, $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a Mercer kernel and $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$. A QUBO formulation of the 2-means problem is given by*

$$\min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{Q}\boldsymbol{z}, \quad \boldsymbol{Q} \coloneqq -\boldsymbol{K} + \operatorname{diag}\left(\mathbf{1}^\top \boldsymbol{K}\right) \; , \tag{3.4}$$

where $\boldsymbol{K}$ is the centered kernel matrix, i. e.,

$$K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j) - \frac{1}{n}\sum_{\boldsymbol{x}\in\mathcal{D}} K(\boldsymbol{x}_i, \boldsymbol{x}) - \frac{1}{n}\sum_{\boldsymbol{x}'\in\mathcal{D}} K(\boldsymbol{x}', \boldsymbol{x}_j) + \frac{1}{n^2}\sum_{\boldsymbol{x},\boldsymbol{x}'\in\mathcal{D}} K(\boldsymbol{x}, \boldsymbol{x}')$$

Proof. We gather the data in a matrix $\boldsymbol{X} \coloneqq (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, and assume that it is centered, i.e., $\boldsymbol{X}\mathbf{1} = \mathbf{0}$. One can show that minimizing the within cluster scatter is equivalent to maximizing the *between cluster*

scatter [150]

$$\max_{C_1,C_2\in\mathcal{D}} |C_1||C_2| \left\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right\|^2 . \tag{3.5}$$

Since $|C_1| + |C_2| = n$, the product $|C_1||C_2|$ is maximized if $|C_1| = |C_2| = \frac{n}{2}$. With assuming $|C_1| \approx |C_2|$, we obtain

$$|C_1||C_2| \left\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right\|^2 \approx \frac{n^2}{4} \left\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right\|^2 = \left\|\frac{n}{2}\left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right)\right\|^2 \approx \left\||C_1|\boldsymbol{\mu}_1 - |C_2|\boldsymbol{\mu}_2\right\|^2 .$$

We introduce two binary vectors $\boldsymbol{z}_1, \boldsymbol{z}_2 \in \mathbb{B}^n$, with the i-th entry indicating whether data point $\boldsymbol{x}_i$ is an element of cluster C_1 or C_2, respectively. That is, the i-th entry of $\boldsymbol{z}_1$ ($\boldsymbol{z}_2$) is 1 if $\boldsymbol{x}_i \in C_1$ ($\boldsymbol{x}_i \in C_2$) and 0 otherwise. It holds that $|C_1|\boldsymbol{\mu}_1 = \boldsymbol{X}\boldsymbol{z}_1$ and $|C_2|\boldsymbol{\mu}_2 = \boldsymbol{X}\boldsymbol{z}_2$ and thus $\left\||C_1|\boldsymbol{\mu}_1 - |C_2|\boldsymbol{\mu}_2\right\|^2 = \left\|\boldsymbol{X}(\boldsymbol{z}_1 - \boldsymbol{z}_2)\right\|^2$. With $\boldsymbol{z}_1 = \mathbf{1} - \boldsymbol{z}_2$ Equation (3.5) can be rewritten to

$$\max_{\boldsymbol{z}\in\mathbb{B}^n} \left\|\boldsymbol{X}(2\boldsymbol{z} - \mathbf{1})\right\|^2 \Leftrightarrow \max_{\boldsymbol{z}\in\mathbb{B}^n} 4(\boldsymbol{X}\boldsymbol{z})^\top \boldsymbol{X}\boldsymbol{z} - 4(\boldsymbol{X}\mathbf{1})^\top \boldsymbol{X}\boldsymbol{z} \Leftrightarrow \min_{\boldsymbol{z}\in\mathbb{B}^n} -\boldsymbol{z}^\top \boldsymbol{K}\boldsymbol{z} + \mathbf{1}^\top \boldsymbol{K}\boldsymbol{z} .$$

with $\boldsymbol{K} = \boldsymbol{X}^\top \boldsymbol{X}$ being the kernel matrix of the linear kernel. With similar arguments, one can use arbitrary kernel matrices. □

The QUBO formulation in Equation (3.4) adheres a certain type of symmetry, since it holds that $E_{\boldsymbol{Q}}(\boldsymbol{z}) = E_{\boldsymbol{Q}}(\mathbf{1} - \boldsymbol{z})$. This ambiguity can be removed by fixing a single entry to zero or one, w.l.o.g. $z_n = 0$. An equivalent formulation to Equation (3.4) is then given by

$$\min_{\boldsymbol{z}\in\mathbb{B}^{n-1}} \boldsymbol{z}^\top \boldsymbol{Q}_{[n-1],[n-1]}\boldsymbol{z}, \quad \boldsymbol{Q} \coloneqq -2\boldsymbol{K}_{[n-1],[n-1]} + \operatorname{diag}\left(\mathbf{1}^\top \boldsymbol{K}_{[n],[n-1]}\right) .$$

3.2.4 Vector Quantization

We first present a VQ QUBO-embedding based on the k-medoids objective.

Proposition 3.4 (KMEDVQ [58])**.** *Let $n, k \in \mathbb{N}$, $k \leq n$, $D : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a distance measure, $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$ and $\boldsymbol{D}$ be the corresponding distance matrix, i. e., $D_{ij} = D(\boldsymbol{x}_i, \boldsymbol{x}_j)$. A QUBO formulation of VQ w.r.t. to the k-medoids objective is given by*

$$\min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{Q}\boldsymbol{z}, \quad \boldsymbol{Q} \coloneqq -\alpha\boldsymbol{D} + \lambda\mathbf{1}\mathbf{1}^\top + \operatorname{diag}(\beta\boldsymbol{D}\mathbf{1} - 2\lambda k\mathbf{1}) , \tag{3.6}$$

where the parameters $\alpha, \beta \in \mathbb{R}_+$ balance the objectives of finding central and far apart data points. The parameter $\lambda \in \mathbb{R}_+$ guarantees that exactly k medoids are chosen.

Proof. The formulation is based on the observation that k-medoids aims to minimize the within cluster scatter and equivalently maximize the between cluster scatter. Minimizing the within cluster scatter can be written as selecting the k most central data points

$$\min_{\mathcal{W}\subset\mathcal{D}} \sum_{\boldsymbol{x}\in\mathcal{W}} \sum_{\boldsymbol{x}'\in\mathcal{D}} D(\boldsymbol{x}, \boldsymbol{x}')$$
$$\text{s.t. } |\mathcal{W}| = k ,$$

while maximizing the between cluster scatter is similar to finding far apart data points

$$\begin{aligned} \max_{\mathcal{W}\subset\mathcal{D}} \sum_{\boldsymbol{x}\in\mathcal{W}} \sum_{\boldsymbol{x}'\in\mathcal{W}} D(\boldsymbol{x},\boldsymbol{x}') \\ \text{s.t. } |\mathcal{W}| = k \ . \end{aligned}$$

Putting both objectives together by weighting them with parameters $\alpha, \beta > 0$

$$\begin{aligned} \min_{\mathcal{W}\subset\mathcal{D}} \ -\alpha \sum_{\boldsymbol{x}\in\mathcal{W}} \sum_{\boldsymbol{x}'\in\mathcal{W}} D(\boldsymbol{x},\boldsymbol{x}') + \beta \sum_{\boldsymbol{x}\in\mathcal{W}} \sum_{\boldsymbol{x}'\in\mathcal{D}} D(\boldsymbol{x},\boldsymbol{x}') \\ \text{s.t. } |\mathcal{W}| = k \ . \end{aligned}$$

Since $\mathcal{W} \subset \mathcal{D}$, we can indicate the membership of the data point $\boldsymbol{x}_i$ in $\mathcal{W}$ by a binary variable z_i. This leads to a QBP formulation by using the distance matrix $\boldsymbol{D}$

$$\begin{aligned} \min_{\boldsymbol{z}\in\mathbb{B}^n} \ -\alpha \boldsymbol{z}^\top \boldsymbol{D}\boldsymbol{z} + \beta \left(\boldsymbol{D}\mathbf{1}\right)^\top \boldsymbol{z} \\ \text{s.t. } \mathbf{1}^\top \boldsymbol{z} = k \ . \end{aligned}$$

Using Proposition 3.1, we obtain the QUBO formulation

$$\min_{\boldsymbol{z}\in\mathbb{B}^n} \ -\alpha \boldsymbol{z}^\top \boldsymbol{D}\boldsymbol{z} + \beta \left(\boldsymbol{D}\mathbf{1}\right)^\top \boldsymbol{z} + \lambda \left(\boldsymbol{z}^\top \mathbf{1}\mathbf{1}^\top \boldsymbol{z} - 2k\mathbf{1}^\top \boldsymbol{z}\right) \ ,$$

with a suitable $\lambda > 0$. □

Depending on the data at hand, one can use an appropriate distance measure. Typically, the euclidean distance $\|\boldsymbol{x} - \boldsymbol{y}\|$ is used, but one can also fall back on more robust measures such as *Welsch's M-estimator* [151]

$$D(\boldsymbol{x},\boldsymbol{y}) = 1 - \exp\left(\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2\right) \ . \tag{3.7}$$

We also present a QUBO formulation for the MD-VQ objective (Definition 2.14)

$$\min_{\mathcal{W}} \ \|\boldsymbol{\phi}_{\mathcal{D}} - \boldsymbol{\phi}_{\mathcal{W}}\|^2 \tag{3.8a}$$

$$\text{s.t. } |\mathcal{W}| = k \ , \tag{3.8b}$$

where $\boldsymbol{\phi}_{\mathcal{W}} = \frac{1}{k}\sum_{\boldsymbol{x}\in\mathcal{W}} \phi(\boldsymbol{x})$ and $\boldsymbol{\phi}_{\mathcal{D}} = \frac{1}{n}\sum_{\boldsymbol{x}\in\mathcal{D}} \phi(\boldsymbol{x})$ denote the mean vectors in some feature space for a feature map $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$.

Proposition 3.5 (MDVQ [83]). *Let $n, k \in \mathbb{N}$, $k \leq n$, $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a Mercer kernel, $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$ and $\boldsymbol{K}$ be the corresponding kernel matrix. A QUBO formulation for MD-VQ is given by*

$$\min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{Q}\boldsymbol{z}, \quad \boldsymbol{Q} := \frac{1}{k^2}\boldsymbol{K} + \lambda \mathbf{1}\mathbf{1}^\top - 2\,\mathrm{diag}\left(\frac{1}{nk}\boldsymbol{K}\mathbf{1} + \lambda k \mathbf{1}\right) \ , \tag{3.9}$$

for a large enough $\lambda > 0$.

Proof. We rewrite the MD in terms of inner products

$$\begin{aligned}\min_{\mathcal{W}} \|\boldsymbol{\phi}_{\mathcal{D}} - \boldsymbol{\phi}_{\mathcal{W}}\|^2 &\Leftrightarrow \min_{\mathcal{W}} \boldsymbol{\phi}_{\mathcal{W}}^\top \boldsymbol{\phi}_{\mathcal{W}} - 2\boldsymbol{\phi}_{\mathcal{D}}^\top \boldsymbol{\phi}_{\mathcal{W}} \\ &\Leftrightarrow \min_{\mathcal{W}} \frac{1}{k^2} \sum_{\boldsymbol{x}\in\mathcal{W}} \sum_{\boldsymbol{x}'\in\mathcal{W}} K(\boldsymbol{x},\boldsymbol{x}') - \frac{2}{kn} \sum_{\boldsymbol{x}\in\mathcal{W}} \sum_{\boldsymbol{x}'\in\mathcal{D}} K(\boldsymbol{x},\boldsymbol{x}') \ ,\end{aligned}$$

with $K(\boldsymbol{x},\boldsymbol{x}') = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}')$ since K is a Mercer kernel. Letting the binary variable z_i indicate whether $\boldsymbol{x}_i$ is an element of $\mathcal{W} \subset \mathcal{D}$, we obtain a QBP formulation equivalent to Equation (3.8)

$$\begin{aligned}\min_{\boldsymbol{z}\in\mathbb{B}^n} \ & \frac{1}{k^2} \boldsymbol{z}^\top \boldsymbol{K} \boldsymbol{z} - \frac{2}{nk} (\boldsymbol{K}\mathbf{1})^\top \boldsymbol{z} \\ \text{s.t. } & \mathbf{1}^\top \boldsymbol{z} = k \ .\end{aligned}$$

Again, using Proposition 3.1, we obtain a QUBO

$$\min_{\boldsymbol{z}\in\mathbb{B}^n} \ \frac{1}{k^2} \boldsymbol{z}^\top \boldsymbol{K} \boldsymbol{z} - \frac{2}{nk} (\boldsymbol{K}\mathbf{1})^\top \boldsymbol{z} + \lambda \boldsymbol{z}^\top \mathbf{1}\mathbf{1}^\top \boldsymbol{z} - 2\lambda k \boldsymbol{z} \ ,$$

for large enough $\lambda > 0$. □

Interestingly, there is a connection between KMEDVQ and MDVQ.

Proposition 3.6. *Let $n, k \in \mathbb{N}$, $k \leq n$, $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a Mercer kernel, $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$, $k < n$ and $\boldsymbol{K}$ be the corresponding kernel matrix. Defining the matrix $\boldsymbol{D} := \mathbf{1}\mathbf{1}^\top - \boldsymbol{K}$, the QUBO formulations in Proposition 3.4 and Proposition 3.5 are equivalent if we set $\alpha = \frac{1}{k^2}$, $\beta = \frac{1}{kn}$ and $\lambda = \lambda' + \frac{1}{k^2}$, where λ is the penalty parameter for linear equality constraint from Proposition 3.4 and λ' from Proposition 3.5.*

Proof. Inserting α, β and λ into the QUBO matrix definition in Proposition 3.4 leads to

$$\begin{aligned}& -\alpha\boldsymbol{D} + \lambda\mathbf{1}\mathbf{1}^\top + \operatorname{diag}(\beta\boldsymbol{D}\mathbf{1} - 2\lambda k\mathbf{1}) \\ \Leftrightarrow\ & -\alpha(\mathbf{1}\mathbf{1}^\top - \boldsymbol{K}) + \lambda\mathbf{1}\mathbf{1}^\top + \operatorname{diag}(\beta(\mathbf{1}\mathbf{1}^\top - \boldsymbol{K})\mathbf{1} - 2\lambda k\mathbf{1}) \\ \Leftrightarrow\ & \alpha\boldsymbol{K} + (\lambda - \alpha)\mathbf{1}\mathbf{1}^\top + \operatorname{diag}(-\beta\boldsymbol{K}\mathbf{1} - (2\lambda k - n\beta)\mathbf{1}) \\ \Leftrightarrow\ & \frac{1}{k^2}\boldsymbol{K} + \lambda'\mathbf{1}\mathbf{1}^\top + \operatorname{diag}\left(-\frac{2}{nk}\boldsymbol{K}\mathbf{1} - \left(2\left(\lambda' + \frac{1}{k^2}\right)k - 2\frac{n}{nk}\right)\mathbf{1}\right) \\ \Leftrightarrow\ & \frac{1}{k^2}\boldsymbol{K} + \lambda'\mathbf{1}\mathbf{1}^\top - 2\operatorname{diag}\left(\frac{1}{nk}\boldsymbol{K}\mathbf{1} + \lambda' k\mathbf{1}\right) \ ,\end{aligned}$$

which corresponds to the QUBO matrix in Proposition 3.5. □

Using Proposition 3.6, we can see that MDVQ is a special case of KMEDVQ by choosing parameters in a specific way and using distance measures of the form $D(\boldsymbol{x},\boldsymbol{y}) = 1 - K(\boldsymbol{x},\boldsymbol{y})$ for some Mercer kernel. With the commonly used RBF kernel with a bandwidth of 2, one exactly recovers Welsch's M-estimator Equation (3.7). In fact, there is an interesting relationship between specific kernels and the euclidean distance in the feature space.

Proposition 3.7. *Let $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a Mercer kernel and $\phi : \mathbb{X} \to \mathbb{R}^d$ the underlying feature map, that is $\phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}') = K(\boldsymbol{x}, \boldsymbol{x}')$. If all feature vectors lie on the unit sphere, i. e., $\phi(\boldsymbol{x}) = 1$, $\forall \boldsymbol{x} \in \mathbb{X}$, then we can obtain a distance measure $D : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ by*

$$D(\boldsymbol{x}, \boldsymbol{x}') = 1 - K(\boldsymbol{x}, \boldsymbol{x}') \ . \tag{3.10}$$

Proof. Consider the euclidean distance between feature vectors $\phi(\boldsymbol{x})$ and $\phi(\boldsymbol{x}')$

$$\left\|\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')\right\|^2 = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}) + \phi(\boldsymbol{x}')^\top \phi(\boldsymbol{x}') - 2\phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}') = 2 - 2K(\boldsymbol{x}, \boldsymbol{x}') \ .$$

Dividing the RHS by 2, we obtain the claim. □

With having different ML QUBO formulations at hand, we go over to a QUBO property which largely effects the performance of QO, especially its runtime.

3.3 Solvability in Terms of Spectral Gap

As we have seen in Section 2.4.2, we need an adiabatic evolution of the quantum system to guarantee finding an optimal solution. The speed at which the Hamiltonian can safely evolve without the system leaving its ground state depends on the optimum SG, which is the minimal difference between the two lowest eigenvalues over time. For a Hamiltonian $\boldsymbol{H}$ with lowest eigenvalue $E^1_{\boldsymbol{H}}$ and second to lowest eigenvalue $E^2_{\boldsymbol{H}}$ it is given by $\gamma_{\boldsymbol{H}} := E^2_{\boldsymbol{H}} - E^1_{\boldsymbol{H}}$. A small SG requires a slow change rate, which leads to a long, potentially exponential (c.f. [105]) *annealing time*. It is therefore desirable to somehow increase the SG by choosing the initial Hamiltonian $\boldsymbol{H}_\mathrm{I}$ or the problem Hamiltonian $\boldsymbol{H}_\mathrm{P}$ appropriately. In QA hardware, such as the one provided by D-Wave, $\boldsymbol{H}_\mathrm{I}$ is usually prescribed and the only free variable is $\boldsymbol{H}_\mathrm{P}$.

To bridge the gap to QUBO, it is important to make the distinction between the parameter matrix $\boldsymbol{Q}$ and its corresponding Hamiltonian $\boldsymbol{H}_{\boldsymbol{Q}}$: The entries along the diagonal of the latter correspond to the values $\boldsymbol{z}^\top \boldsymbol{Q} \boldsymbol{z}$ for every possible $\boldsymbol{z} \in \mathbb{B}^n$. Therefore, the SG of $\boldsymbol{H}_{\boldsymbol{Q}}$ is simply the difference between the lowest and second-to-lowest values of $E_{\boldsymbol{Q}}$ (which is very hard to compute for large n), i. e.,

$$\gamma_{\boldsymbol{H}_{\boldsymbol{Q}}} = \min_{\boldsymbol{z} \in \mathbb{B}^n \setminus \mathcal{Z}^*(\boldsymbol{Q})} E_{\boldsymbol{Q}}(\boldsymbol{z}) - \min_{\boldsymbol{z} \in \mathbb{B}^n} E_{\boldsymbol{Q}}(\boldsymbol{z}) \ . \tag{3.11}$$

We thus use the terms optimum energy gap and SG interchangeably, the eigenvalues of $\boldsymbol{Q}$ hold no particular relevance.

3.3.1 Bounding the Spectral Gap

We recall from Section 2.4.2, that the time-dependent Hamiltonian during the adiabatic process is given by $\boldsymbol{H}(s) = A(s)\boldsymbol{H}_\mathrm{I} + B(s)\boldsymbol{H}_\mathrm{P}$, where $A, B : [0, 1] \to [0, 1]$ are monotonically decreasing and increasing, respectively, and $A(0) = B(1) = 1$, $A(1) = B(0) = 0$, i. e., $\boldsymbol{H}(0) = \boldsymbol{H}_\mathrm{I}$ and $\boldsymbol{H}(1) = \boldsymbol{H}_\mathrm{P}$. Due to the adiabatic theorem, the time T needed scales as

$$T \in \mathcal{O}\left(\max_{s \in [0,1]} \frac{1}{\gamma^2_{\boldsymbol{H}(s)}}\right) \ ,$$

but the minimal SG $\min_{s\in[0,1]} \gamma_{\boldsymbol{H}(s)}$ cannot be easily predicted from either $\gamma_{\boldsymbol{H}_\mathrm{I}}$ or $\gamma_{\boldsymbol{H}_\mathrm{P}}$. However, we can still make some statements about it using known results about eigenvalues of Hermitian matrices.

Theorem 3.1 (Weyl). *[152]] Let $\boldsymbol{M}, \boldsymbol{N}, \boldsymbol{R}$ be $m \times m$ Hermitian matrices with $\boldsymbol{N} + \boldsymbol{R} = \boldsymbol{M}$. Let μ_i, ν_i, ρ_i denote their respective eigenvalues in ascending order, i.e., $\mu_i \leq \mu_{i+1}$ $\forall 1 \leq i < m$, and for ν_i, ρ_i analogously. Then* Weyl's inequality *holds for all $1 \leq i \leq m$:*

$$\nu_i + \rho_1 \leq \mu_i \leq \nu_i + \rho_m \ . \tag{3.12}$$

We can use Weyl's inequality to deduce an upper bound on $\min_{s\in[0,1]} \gamma_{\boldsymbol{H}(s)}$.

Proposition 3.8. *Let $\boldsymbol{H}_\mathrm{P}$, $\boldsymbol{H}_\mathrm{I}$, $\boldsymbol{H}(s)$ and $A(s)$ be defined as before and let $B(s) = 1 - A(s)$. Then*

$$\min_{s\in[0,1]} \gamma_{\boldsymbol{H}(s)} \leq \gamma_{\boldsymbol{H}_\mathrm{P}} \ . \tag{3.13}$$

Proof. Applying Weyl's inequality multiple times we find the following bounds on the SG of sums of two Hamiltonians:

$$\underbrace{\mu_2 - \mu_1}_{=\gamma_{\boldsymbol{M}}} \leq \underbrace{\nu_2 - \nu_1}_{=\gamma_{\boldsymbol{N}}} + \underbrace{(\rho_m - \rho_1)}_{:=\Gamma(\boldsymbol{R})} \ . \tag{3.14}$$

Since $A(s)$ is monotone, we obtain

$$\min_{s\in[0,1]} aA(s) + bB(s) = \min_{s\in[0,1]} aA(s) + b\,(1 - A(s)) = \min\{a, b\} \ .$$

Applying Equation (3.14) to $\min_{s\in[0,1]} \gamma_{\boldsymbol{H}(s)}$ leads to

$$\begin{aligned}
\min_{s\in[0,1]} \gamma_{\boldsymbol{H}(s)} &\leq \min_{s\in[0,1]} A(s)\Gamma(\boldsymbol{H}_\mathrm{I}) + B(s)\gamma_{\boldsymbol{H}_\mathrm{P}} && (3.15)\\
&= \min_{s\in[0,1]} A(s)\left(\Gamma(\boldsymbol{H}_\mathrm{I}) - \gamma_{\boldsymbol{H}_\mathrm{P}}\right) + \gamma_{\boldsymbol{H}_\mathrm{P}} && (3.16)\\
&= \min\{\gamma_{\boldsymbol{H}_\mathrm{P}}, \Gamma(\boldsymbol{H}_\mathrm{I})\} \leq \gamma_{\boldsymbol{H}_\mathrm{P}} \ , && (3.17)
\end{aligned}$$

concluding the proof.

□

This result provides motivation to increase $\gamma_{\boldsymbol{H}_\mathrm{P}}$ when trying to improve QO performance, as it is an upper bound on $\gamma_{\boldsymbol{H}(s)}$: Increasing it does not guarantee a larger minimal SG, but is a necessary precondition. Finding a lower bound is generally very hard.

3.3.2 QUBO Formulation for Spectral Gap

Computing the SG of a QUBO is as hard as computing the optimal solution itself. Interestingly, we can find a QBP formulation for obtaining the SG of a given QUBO instance.

Proposition 3.9 (SG-QBP). *Let $\boldsymbol{Q} \in \mathcal{Q}_n$ represent the matrix of a QUBO instance with a unique*

Algorithm 1 SPECTRALGAPQUBO

Input: QUBO instance $\boldsymbol{Q} \in \mathcal{Q}_n$
Output: Spectral gap $\gamma_{\boldsymbol{H_Q}}$

1: $v \leftarrow \infty$
2: **for** $i = 1$ to n **do**
3: $\quad v_0^i \leftarrow \min_{\boldsymbol{z} \in \mathbb{B}^n_{i \leftarrow 1}} E_{\boldsymbol{Q}}(\boldsymbol{z})$
4: $\quad v_1^i \leftarrow \min_{\boldsymbol{z} \in \mathbb{B}^n_{i \leftarrow 0}} E_{\boldsymbol{Q}}(\boldsymbol{z})$
5: $\quad v \leftarrow \min\{v, v_0^i + v_1^i\}$
6: **end for**
7: $\gamma_{\boldsymbol{H_Q}} = v - 2 \min_{\boldsymbol{z} \in \mathbb{B}^n} E_{\boldsymbol{Q}}(\boldsymbol{z})$

optimizer, that is $|\mathcal{Z}^*(\boldsymbol{Q})| = 1$. *Further, let*

$$v := \min_{\boldsymbol{z}, \boldsymbol{z}' \in \mathbb{B}^n} E_{\boldsymbol{Q}}(\boldsymbol{z}) + E_{\boldsymbol{Q}}(\boldsymbol{z}') \tag{3.18a}$$

$$\text{s.t. } \boldsymbol{z} \neq \boldsymbol{z}' \tag{3.18b}$$

Then

$$\gamma_{\boldsymbol{H_Q}} = v - 2 \min_{\boldsymbol{z} \in \mathbb{B}^n} E_{\boldsymbol{Q}}(\boldsymbol{z}) \tag{3.19}$$

Proof. Follows from Equation (3.11). □

The constraint in Equation (3.18b) can be enforced iteratively by fixing the variables z_i and z_i' to different values, that is $z_i = 1 - z_i'$. An overview of this procedure is given in Algorithm 1. We recall from Chapter 2 that $\mathbb{B}^n_{\{i\} \leftarrow b}$ denotes the set of binary vectors, whose i-th entry is fixed to the value $b \in \mathbb{B}$. For obtaining equivalence to Equation (3.18), we thus can solve $2n + 1$ QUBO problems appearing in Algorithm 1. Exact solutions are of course intractable to compute, but as we will see in Chapter 4, bounds on the optimal QUBO value can be computed efficiently. Thus, we can obtain efficiently computable bounds on the SG for general QUBO problems.

3.4 Experimental Evaluation

We have seen in Sections 3.2.2 and 3.2.4 that linear equality constraints have to be integrated into the objective function, for obtaining a QUBO problem. This can be done with using positive penalty parameters to punish a violation of the constraints, as is evident from Proposition 3.1. It is well known, that choosing larger penalty parameters also results in a larger SG [146]. We can attribute problem properties to effects on the SG of the resulting Hamiltonian of the QUBO formulation. We thus conduct an empirical evaluation of QUBO formulations and their properties. For each experiment, the steps we take are as follows: (i) Choose problem type and hyperparameters, (ii) Sample dataset with known properties, (iii) Compute QUBO parameters and record the SG. Using the acquired data, we investigate the relationship between data parameters and SG, which has a high impact on problem hardness for QC, as we have shown before. We consider the two problems of binary clustering and SVM training, which we described in previous sections.

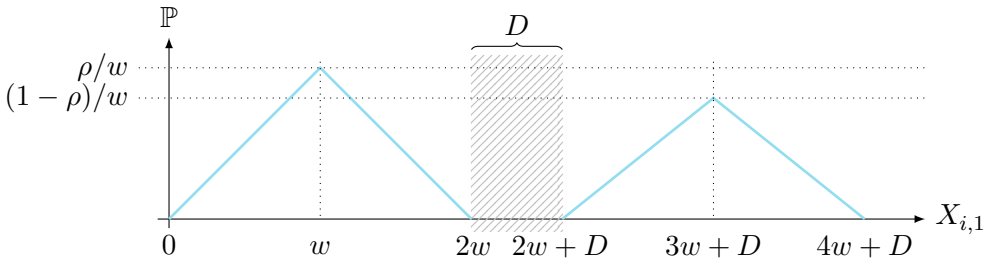


Figure 3.2: Distribution of the first dimension of the 2-dimensional synthetic data used for our experiments (before applying the rotation): Two clusters are sampled such that there is a separating margin of at least size D between them. The parameter w controls the spread of data points, while r is the ratio between the number of data points in the first vs. the second cluster.

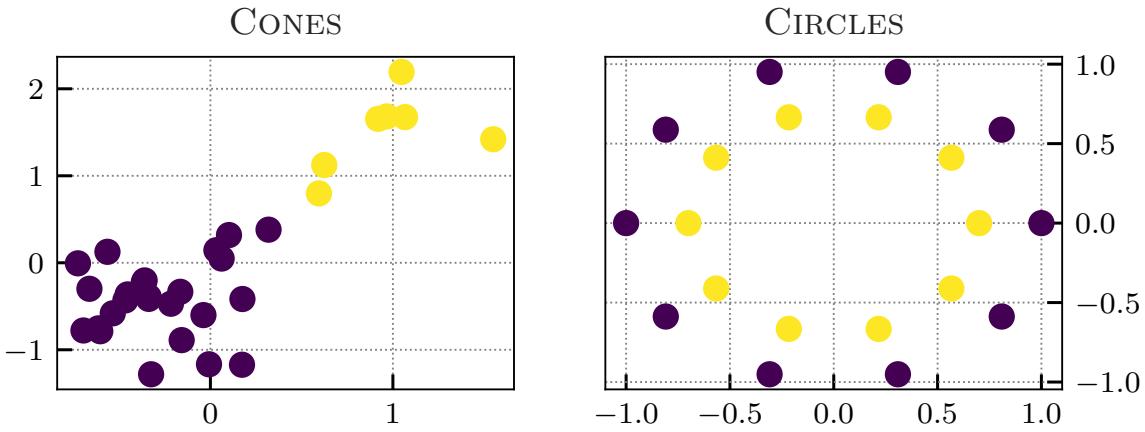


Figure 3.3: Exemplary instances of the datasets used for our experiments.

3.4.1 Data Setup

As input data, we sample synthetic datasets for each repetition of the experiments. To this end, we consider two different types, CONES and CIRCLES, both of which have parameters allowing us to vary the resulting optimization problems' difficulty by adjusting the data class separation.

CONES Let $n \in \mathbb{N}$ with $n \geq 2$ denote the number of data points, $\rho \in (0, 1)$ the cluster size ratio, $w > 0$ a spread parameter, and $D \in \mathbb{R}$ a separating margin size. We set $n_1 := \min\{1, \lfloor \rho n \rfloor\}$ and $n_2 := 1 - n_1$ as the cluster sizes. We create a matrix $\boldsymbol{X} \in \mathbb{R}^{n \times 2}$ where every entry X_{ij} is sampled i.i.d. from a triangular distribution within the interval $[0, 2w]$ and with mode w. We chose the triangular distribution over a normal distribution because it has no outliers, which allows us to define a hard lower bound on the separating margin between clusters. For all $i > n_1$ we then set $X_{i,1} \mapsto X_{i,1} + 2w + D$, which shifts all of these points such that $\|\boldsymbol{X}_{i,\cdot} - \boldsymbol{X}_{\ell,\cdot}\|_2 \geq D$ is tight for all $i \in [n_1]$ and $n_1 < \ell \leq n$. The distribution of $\boldsymbol{X}_{\cdot,1}$ is visualized in Figure 3.2; the distribution of $\boldsymbol{X}_{\cdot,2}$ consists of just a single triangle from 0 to $2w$ with height $1/w$ at mode w. Next, we sample θ uniformly from $[0, 2\pi)$ and apply $\boldsymbol{X} \mapsto \boldsymbol{X}\boldsymbol{R}(\theta)$, where $\boldsymbol{R}(\theta)$ is a 2D rotation matrix. This rotation leaves the distances unchanged but

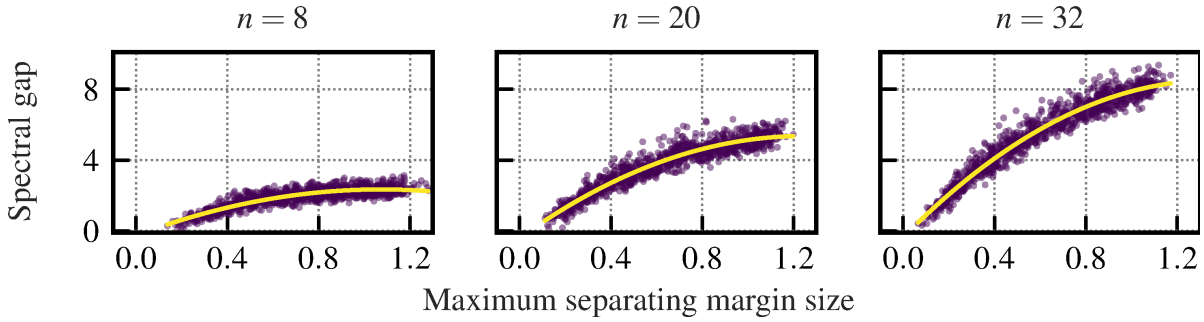


Figure 3.4: SG of QUBO instances according to Proposition 3.3 against maximum separating margin size D for CONES; $w = 0.2$, $\rho = 0.5$ fixed, 1000 random datasets with $n \in \{8, 20, 32\}$ and $D \in [0, 1]$ uniformly sampled. The yellow curve is a fitted quadratic function.

introduces another degree of freedom. Lastly, we center the data by computing $\boldsymbol{\mu}_j \coloneqq \sum_{i=1}^{n} X_{ij}/n$ and applying $X_{ij} \mapsto X_{ij} - \boldsymbol{\mu}_j$ for all $i, j \in [n] \times [2]$. The target vector $\boldsymbol{y} \in \mathbb{S}^n$ is set to $y_i = -1$ for $i \in [n_1]$ and $y_i = +1$ for $n_1 < i \leq n$.

CIRCLES As a second dataset type, we consider two circles, which are not linearly separable in $\mathbb{R}^2$. The radius of the outer circle is fixed to 1 and for our experiments we vary the radius of the inner circle r. The circles consist of an equal number of points $n/2$ and Gaussian noise with standard deviation σ is added to every point (see Figure 3.3, right). To bridge the gap to linear separability, we project the dataset to a higher-dimensional feature space via a feature map $\phi : \mathbb{R}^2 \to \mathbb{R}^3$, $\phi(\boldsymbol{x}) \coloneqq (x_1, x_2, a\,\|\boldsymbol{x}\|^2)^\top$. In this space, the data is linearly separable. A corresponding kernel function is given by $K(\boldsymbol{x}, \boldsymbol{y}) \coloneqq \phi(\boldsymbol{x})^\top \phi(\boldsymbol{y}) = \boldsymbol{x}^\top \boldsymbol{y} + a^2 \|\boldsymbol{x}\|^2 \|\boldsymbol{y}\|^2$. For every experiment, we compare three different problem sizes, that is, we consider $n \in \{8, 20, 32\}$. To make the SG comparable between QUBO instances of the same size, we scale each $\boldsymbol{Q}$ such that $\|\boldsymbol{Q}\|_\infty = \max_{i,j \in [n]} |Q_{ij}| = 1$.

3.4.2 Biclustering

We first explore the biclustering QUBO from Proposition 3.3. Changing the maximum separating margin size between the two clusters changes the SG of the corresponding QUBO instance, as shown in Figures 3.4 and 3.5: For Figure 3.4, we sample 1000 different CONES datasets with varying cluster distances in $[0, 1]$ and fix $w = 0.2$, $\rho = 0.5$. We find that the SG is increasing with an increasing problem size n and that there is a clear quadratic positive correlation between the SG and the margin size.

A similar setup can be found in Figure 3.5, where 1000 different CIRCLES datasets are sampled with varying inner radius in $[0, 1]$ and $\sigma = 0.05$, $\rho = 0.5$. Again we find that the SG increases with an increasing margin size, but with a linear correlation. Since different kernels are used in Figure 3.4 and Figure 3.5, the exact correlation form is dependent on the exact dataset and the used kernel.

In Figure 3.6, the effects of a varying cluster ratio in $\rho \in [0.1, 0.5]$ are depicted for the CONES setup. We again sample 1000 datasets with fixing $w = 0.2$ and $D = 0.5$. A positive correlation becomes evident between cluster ratio and SG. That is, the QUBO problem is easier to solve with QC when the clusters have the same size. The effect that the plots look like a step function for small n is due to the fact that there $n/2$ different configurations, e.g., for $n = 8$, we can have the four cases $n_1 = 1$, $n_1 = 2$,

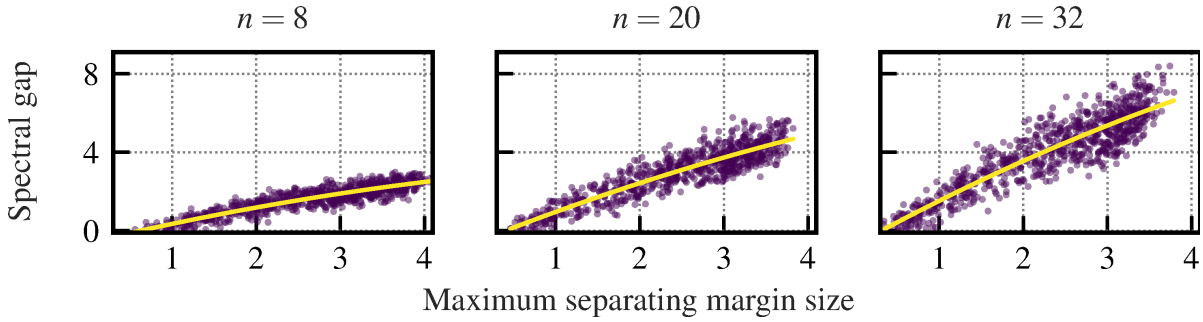


Figure 3.5: SG of QUBO instances according to Proposition 3.3 against maximum separating margin size D for CIRCLES; $\sigma = 0.05$, $\rho = 0.5$ fixed, 1000 random datasets with $n \in \{8, 20, 32\}$ and $r \in [0, 1]$ uniformly sampled. The yellow curve is a fitted quadratic function.

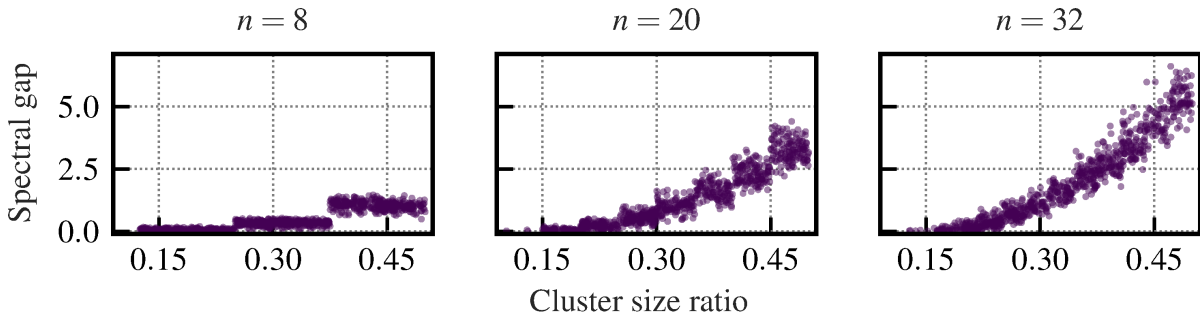


Figure 3.6: SG of QUBO instances according to Proposition 3.3 against cluster ratio for CONES; $D = 0.5$, $w = 0.2$ fixed, 1000 random datasets for $n \in \{8, 20, 32\}$ and $\rho \in [0.1, 0.5]$ uniformly sampled.

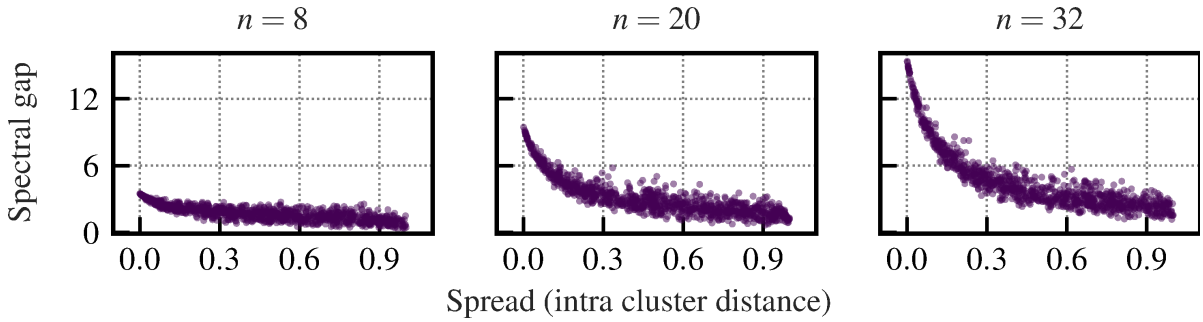


Figure 3.7: SG of QUBO instances according to Proposition 3.3 against spread for CONES; $D = 0.5$, $\rho = 0.5$ fixed, 1000 random datasets for $n \in \{8, 20, 32\}$ and $w \in [0, 1]$ uniformly sampled.

$n_1 = 3$ and $n_1 = 4$.

In Figure 3.7, we vary the spread $w \in [0, 1]$ for 1000 different datasets and fix $D = 0.5$, $\rho = 0.5$. We can see that the SG is negatively correlated to the spread of the dataset.

Putting the results together we can deduce that the SG is positively correlated with the inter-cluster distance or between cluster scatter (separability) and negatively correlated with the intra-cluster distance or within cluster scatter (compactness).

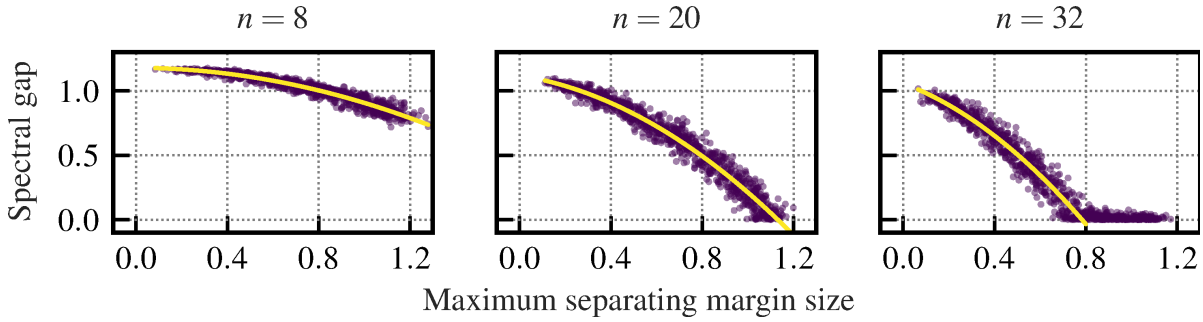


Figure 3.8: SG of QUBO instances according to Proposition 3.2 against maximum separating margin size D for CONES. Same configuration as for Figure 3.4.

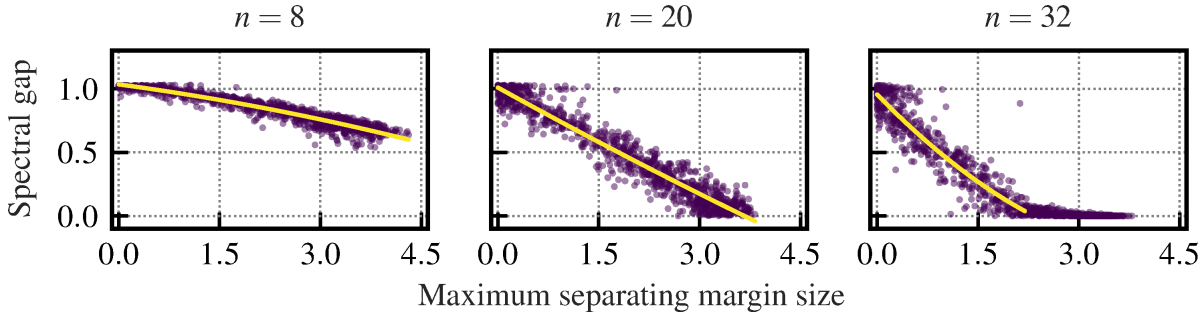


Figure 3.9: SG of QUBO instances according to Proposition 3.2 against maximum separating margin size D for CIRCLES. Same configuration as for Figure 3.5.

3.4.3 Binary Support Vector Machine

We move over to experiments with the SVM QUBO in Proposition 3.2. Again, we depict the effect of changing the maximum separating margin size between the two clusters on the SG of the corresponding QUBO in Figures 3.8 and 3.9. We use the same parameters for the datasets as in Figures 3.4 and 3.5.

Interestingly, we now observe a negative correlation between the SG and the margin size, making the problem harder to solve with a quantum computer if the data is well separable. For $n = 32$ the SG basically vanishes from a certain margin size for CIRCLES. Furthermore, we again observe that this correlation is quadratic for CONES and linear for CIRCLES, leading to a large dependence on the used kernel function and the dataset at hand.

Note that we are considering a QUBO formulation for a soft-margin SVM: even though the SG might be very small, the second best solution might also be satisfactory for solving the original problem. In contrast, the second best solution of a biclustering QUBO is much worse: changing a single bit leads to a data point within the wrong cluster, which increases the energy more dramatically the further the two clusters are separated.

In Figures 3.10 and 3.11, we show the effect of varying λ and C on the SG for CONES. We fix $\rho = 0.5$, $w = 0.2$, $D = 0.5$ and sample 10 000 datasets with $C \in [0, 0.1]$, $\lambda \in [0, 100]$ for Figure 3.10 and $\lambda \in [0, 10]$ for Figure 3.11, respectively. It is evident from Figure 3.10 that the SG decreases as λ and C increase. However, there are interesting intervals for λ when fixing C, such that the SG first increases and then decreases with increasing λ, forming a triangular shape when plotted, which gets more pointy with an increasing value of C – see Figure 3.11 for a closer view. We observe a similar

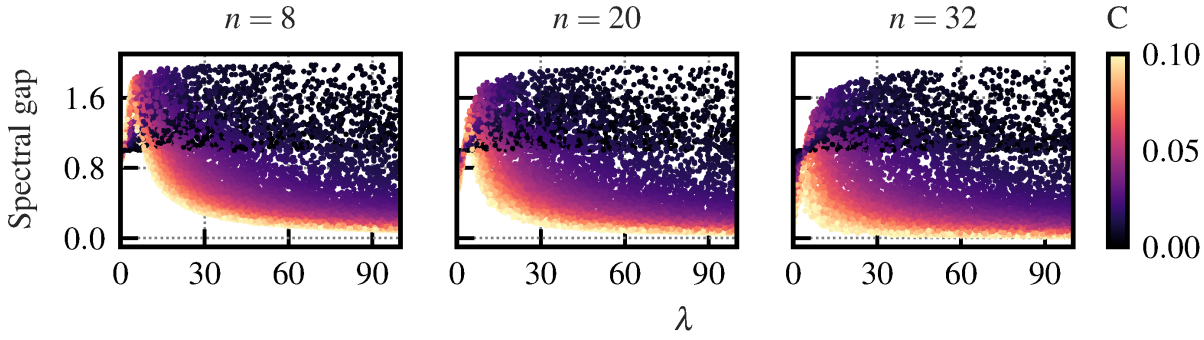


Figure 3.10: SG of QUBO instances according to Proposition 3.2 against λ and C for CONES; $w = 0.2$, $\rho = 0.5$ fixed, 10 000 random datasets for $n \in \{8, 20, 32\}$ and $\lambda \in [0, 100]$, $C \in [0, 0.1]$ uniformly sampled.

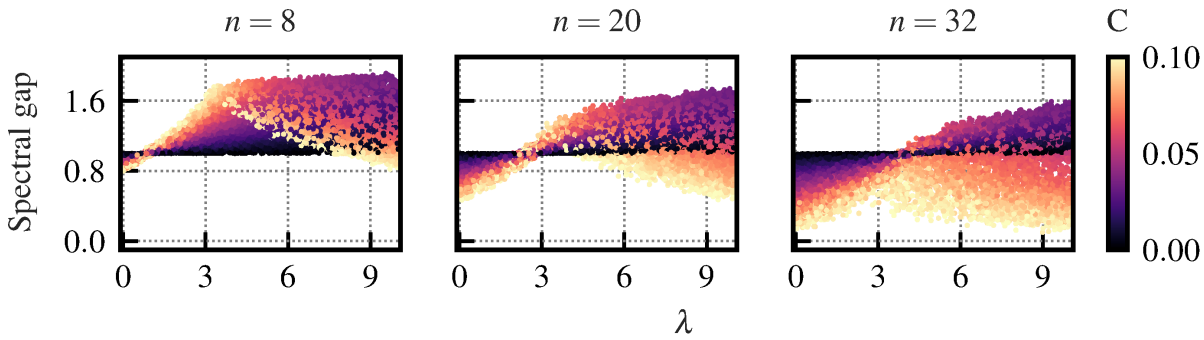


Figure 3.11: Same as Figure 3.10, zoomed in on $\lambda \in [0, 10]$.

effect with CIRCLES.

Combining our observations we deduce that the SG is negatively correlated with the inter-cluster distance (separability) and the parameters λ, C except for a small region.

3.5 Conclusion

In this chapter, we investigated the connection between the problem hardness of classical ML problems—related to the complexity of the underlying data—and their solvability on quantum hardware. We considered QUBO formulations for biclustering and SVM learning. We highlighted that the SG of these formulations impacted their solvability on quantum hardware, and showed how the SG behaves when adapting the problem parameters, which we underpinned with an empirical study.

We found that for biclustering an easier problem also leads to a better solvability on quantum computers. Here, "easy" refers to the separability and compactness of the different classes. We found a positive correlation between these properties and the SG of the corresponding QUBO. Interestingly, this is not the case for the SVM problem, where we would have expected a better solvability for problems with a large separation of the classes. However, we found a negative correlation between separation and SG. Furthermore, other hyperparameters, such as the one controlling the softness of the margin that avoids overfitting, are negatively correlated to the SG. This is due to the balancing of different objectives in one single QUBO problem. Combining these two insights, we conclude that the original problems'

hardness is not directly connected to the solvability on quantum computers, contrary to what one might assume. Instead, it depends not only on the dataset at hand, but also on specifics in the used QUBO formulation.

Another interesting research direction would be to compare the properties of more QUBO formulations of different problems. Furthermore, investigating the effect of the QUBO parameters and not only the problem parameters is an interesting research direction [45]. This can strengthen our intuition about which problems are hard for quantum computers in particular, and the potential and limitations of QC in general.

In the next chapter, we expand our analysis beyond the SG, recognizing that it is just one measure of solvability and the time required to find an optimal solution. While a large SG generally implies faster convergence, choosing an evolution time that is too short can lead to false optima—a fundamental limitation in QO. This issue is further exacerbated in NISQ devices, where quantum computations are subject to significant errors. These errors can distort the optimization landscape, increasing the likelihood of converging to incorrect solutions. To address these challenges, the next chapter provides an in-depth exploration of a robust measure of error proneness, strategies for preserving true optima, and techniques for mitigating noise, ultimately improving the reliability of QO methods on quantum hardware.

CHAPTER 4

Mitigating Data Induced Noise

In the last chapter, we explored the impact of data complexity on QO methods, particularly in the context of CO problems relevant to ML. By analyzing the SG of the problem Hamiltonian, we established a direct link between data characteristics and quantum solvability, highlighting its implications for AQC. Our findings demonstrate that while some intuitive relationships hold—such as the effect of cluster separation on clustering problems—others, like the inverse relation in SVM learning, challenge conventional expectations.

This chapter broadens the scope of our analysis beyond the SG, acknowledging that it is only one factor influencing solvability and the time required to reach an optimal solution. In QO, an insufficient evolution time can lead to convergence at false optima, a challenge that becomes even more pronounced in NISQ devices due to inherent computational errors. These errors can distort the optimization process, further increasing the risk of incorrect solutions. To counteract these limitations, we delve into a robust framework for assessing error proneness, explore methods to preserve true optima, and present strategies for mitigating noise, ultimately enhancing the reliability of QO on current quantum hardware. An overview of this process is given in Figure 4.1.

Apart from QC, hardware acceleration in general is a major driving force in the recent advent of AI. Virtually all large-scale AI models rely on hardware-accelerated training via Tensor Processing Units (TPUs), GPUs, or FPGA. A key ingredient of these accelerators is parallelism—a large computation is split into smaller pieces, solved via multiple compute units. Clearly, each compute unit must read its inputs from memory. However, memory bandwidth is limited. Hence, to achieve a large level of parallelism, the input that each compute unit receives must be as small as possible. To this end, model parameters with limited precision, e.g., 16-bit, 8-bit, or even smaller, are considered and special training procedures are employed to directly train models with low-precision parameters [153]. AI accelerators usually rely on parallel implementations of basic linear algebra routines. There is, however, a multitude of AI problems whose inherent computational complexity does not stem from linear algebra operations. Examples include clustering [149], probabilistic inference [154], or feature selection [155].

As of today, solving such ML-related CO problems exactly is out of reach for high-dimensional instances. However, analog devices [67, 75, 76], ASICs [70], FPGAs [71, 72], and quantum computers [33] have recently made promising progress when it comes to solving CO problems. Despite the promise of quantum speedup, currently available devices that claim to perform AQC could not yet be proven to be faster than classical computing resources [156]. One common issue of QUBO hardware solvers, also called Ising machines, is limited physical precision of the matrix entries, as real-world

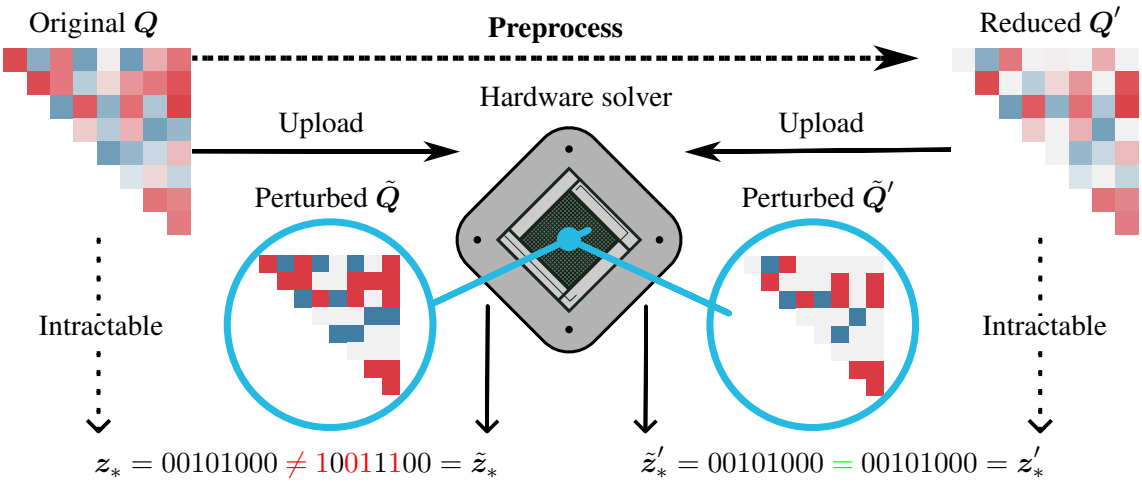


Figure 4.1: Illustration of parameter perturbation for finite precision hardware and the mitigation with our proposed preprocessing method. When we upload the original QUBO $\boldsymbol{Q}$ to the hardware solver, it is perturbed through quantization errors which can lead to spurious optima of the resulting perturbed QUBO $\tilde{\boldsymbol{Q}}$, i.e. $\boldsymbol{z}_* \neq \tilde{\boldsymbol{z}}_*$. Our proposed preprocessing method mitigates this effect by transforming $\boldsymbol{Q}$ into a QUBO $\boldsymbol{Q}'$ which is more robust against hardware errors, while preserving the optimal solution, i.e. $\boldsymbol{z}_* = \boldsymbol{z}'_* = \tilde{\boldsymbol{z}}'_*$.

hardware devices use finite numerical representations. While in theory QUBO is defined with parameters in $\mathbb{R}$, digital devices rely on finite number representations that necessarily sacrifice some precision, as B bits can only encode 2^B distinct values. The well-known solution to this problem in the classical realm is floating-point arithmetic, e.g., IEEE 754. However, it turns out that simply truncating decimal digits of $\boldsymbol{Q}$ is not sufficient, since the resulting optimization problem will have different local and global optima [45] (see Figure 4.1). *Integrated Control Errors* (ICE)[1] pose a similar problem specific to D-Wave's quantum annealers, which randomly distort the Hamiltonian, leading to a skewed energy landscape. Depending on the particular problem instance, this distortion may change the optimum (see Figure 4.1). Classical solutions like IEEE 754 are thus not applicable, due to the physical, analogue representation of the Hamiltonian.

We develop an algorithmic machinery for reducing the numeric precision required to represent QUBO instances, by first having an in-depth look into parameter precision in Section 4.2. The proposed method relies on the DR of any QUBO matrix as a measure of its complexity, which is proportional to the number of bits required to encode the QUBO parameters faithfully: the smaller the DR, the more robust the instance is against distortion. We formalize the notion of optimum preservation between QUBO instances based upon their set of minimizing bit vectors and explore to which extent parameters can be modified without changing said set. Having optimality guarantees at hand, we switch our attention towards reducing the DR of a given QUBO instance in Section 4.3. First, heuristics for reducing the DR by changing single matrix entries are presented. We then formalize the problem of reducing the required precision via an MDP and introduce a fully principled B&B algorithm for exactly solving the problem in a finite number of steps. Efficiently computable bounds are presented for pruning the state space, enhancing our B&B algorithm. Moreover, we explain why greedy methods are likely to produce suboptimal results and propose a combination of B&B and policy rollout [14, 93] to address the greedy

[1] https://docs.dwavesys.com/docs/latest/c_qpu_ice.html (last accessed September 19, 2025)

nature of baseline methods.

For analyzing the effect of data properties on the DR, we conduct an experimental evaluation of ML-related problems in Section 4.4. Similarly to the SG, the DR is also greatly effected by certain properties such as the existence of outliers in the dataset. To evaluate the effectiveness of our proposed heuristics, we first examine randomly generated QUBO instances. We then turn to data-dependent QUBO embeddings, in particular biclustering and VQ (see Chapter 3) and present an embedding for the well-known *subset sum* problem. The performance of our proposed B&B is compared to our developed heuristics and other state-of-the-art methods, indicating the former's superiority. Lastly, we investigate the performance of NISQ devices and FPGA-based Digital Annealing (DA) hardware by using our method for preprocessing QUBO instances. Our results indicate strong benefits not only for mitigating noise in quantum computations but also for error-robustness and possible speed-ups in Ising machines in general.

4.1 Related Work

It is well known that hardware devices tailored towards solving QUBO problems suffer from a limited parameter precision [157]. While FPGA-based DA devices [158] have a fixed *bit-width* for representing problem parameters, e.g., 16-bit, quantum computers are prone to *integrated control errors* [159, 160]. For preserving global optima, in [157] the bit-width is reduced by introducing exponentially many auxiliary variables dependent on the number of reduced bits. This can be combined with the heuristic rounding of the parameters [161], which, however, can lead to different optima. A similar method that respects the underlying hardware topology of a D-Wave quantum annealer can be found in [162]. Instead of enlargening the problem size, [163] follows the approach of solving topologically equivalent instances, each with reduced precision requirements. The number of such instances grows exponentially with the number of reduced bits, and the optimal solution is guaranteed to be preserved only if all instances share the same global optimum.

A more general precision measure than the bit-width, which is only defined for integer parameters, is discussed in [57, 164]. The *maximum coefficient ratio* of a QUBO problem is directly related to the performance on D-Wave quantum annealers. This ratio is largely affected by penalty parameters for incorporating constraints. The authors of [146, 165] try to optimize these penalties by using bounds on the optima of the underlying problem. However, these methods are not applicable for arbitrary QUBO instances, e.g., when the large precision stems from the underlying data and not the problem formulation. In [45], the DR is identified as an improved complexity measure, and a method for iteratively reducing the DR is proposed, which can be applied to any QUBO instance. The underlying idea is to update single QUBO matrix entries within specific interval boundaries, computed by bounding the optimal QUBO value. The method is guaranteed to preserve the original optima. However, the heuristics presented in that work are greedy and often get stuck in local optima. In what follows, we build upon [45] by formulating a more elaborate algorithm that overcomes this issue.

4.2 Parameter Precision

Even though the entries of a QUBO matrix are real-valued in theory, on any real-world computing device there is a limit to the precision with which numbers can be represented. Typically, binary representations are used, where floating-point numbers or 2-complement integers can be represented

with a fixed number of bits. For instance, a register of B bits using 2-complement can represent all integers in $\{-2^{B-1}, \dots, 2^{B-1}-1\}$ (the first bit represents the sign). Values that have a fractional part must be rounded to the nearest integer in order to be represented in 2-complement. Rounding a number $a \in \mathbb{R}$ to the nearest integer is denoted by $\lfloor a \rceil$. By convention, a number exactly halfway between two integers (with fractional part 0.5) is rounded up. Additionally, we write $\lfloor \boldsymbol{A} \rceil$ to denote element-wise rounding for any real-valued matrix $\boldsymbol{A}$. This leads to the important practical observation that any real-world computing device with finite floating-point precision only realizes QUBO instances with integer parameters.

Analog devices also suffer from similar limitations. Even though floating-point precision is often not the bottleneck, unwanted physical effects pose a challenge on the parameter precision. For example, uploading and solving a QUBO problem to D-Wave quantum annealers endures device-specific ICEs. Assume we are given the matrix $\boldsymbol{J} \in \mathbb{R}^{n \times n}$ and the vector $\boldsymbol{h} \in \mathbb{R}^n$ describing the Ising model of the form

$$E_{\boldsymbol{J},\boldsymbol{h}}(\boldsymbol{\sigma}) = \boldsymbol{\sigma}^\top \boldsymbol{J}\boldsymbol{\sigma} + \boldsymbol{h}^\top \boldsymbol{\sigma}, \quad \boldsymbol{\sigma} \in \mathbb{S}^n \ .$$

Instead of accurately representing the energy $E_{\boldsymbol{J},\boldsymbol{h}}(\boldsymbol{\sigma})$ the quantum hardware solves a slightly perturbed problem (see Figure 4.1)

$$E^{\delta}_{\boldsymbol{J},\boldsymbol{h}}(\boldsymbol{\sigma}) = \boldsymbol{\sigma}^\top (\boldsymbol{J} + \delta\boldsymbol{J})\boldsymbol{\sigma} + (\boldsymbol{h} + \delta\boldsymbol{h})^\top \boldsymbol{\sigma}, \quad \boldsymbol{\sigma} \in \mathbb{S}^n \ ,$$

where $\delta\boldsymbol{J}$ and $\delta\boldsymbol{h}$ characterize the errors in the Ising parameters. Such errors appear due to different reasons[2], such as background susceptibility, flux noise of qubits or quantization effects.

4.2.1 Dynamic Range

To quantify the precision required to accurately represent QUBO parameters, we adopt the concept of DR from signal processing [166, 167]. For this, we first introduce the notion of difference sets.

Definition 4.1 (Difference Set)**.** Let $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}$ be finite sets. We define the set of absolute differences between the elements of $\mathcal{X}$ and $\mathcal{Y}$ as

$$D_{\text{set}}(\mathcal{X}, \mathcal{Y}) \coloneqq \{|x - y| : x \in \mathcal{X},\ y \in \mathcal{Y},\ x \neq y\} \ . \tag{4.1}$$

Further, we define $\check{D}(\mathcal{X}, \mathcal{Y}) \coloneqq \min D_{\text{set}}(\mathcal{X}, \mathcal{Y})$ and $\hat{D}(\mathcal{X}, \mathcal{Y}) \coloneqq \max D_{\text{set}}(\mathcal{X}, \mathcal{Y})$ and write $D_{\text{set}}(\mathcal{X}) \equiv D_{\text{set}}(\mathcal{X}, \mathcal{X})$, $\check{D}(\mathcal{X}) \equiv \min D_{\text{set}}(\mathcal{X}, \mathcal{X})$ and $\hat{D}(\mathcal{X}) \equiv \max D_{\text{set}}(\mathcal{X}, \mathcal{X})$ for shorthand.

Definition 4.2 (Dynamic Range)**.** Let $\mathcal{X} \subset \mathbb{R}$ be a finite set. The DR of $\mathcal{X}$ is defined as

$$\mathsf{DR}(\mathcal{X}) \coloneqq \log_2 \left(\frac{\hat{D}(\mathcal{X})}{\check{D}(\mathcal{X})} \right) \ . \tag{4.2}$$

A large DR indicates that many bits are required to represent all elements of $\mathcal{X}$ accurately in binary representation, as the parameters span a wide range of values and require fine gradations. Taking the next larger integer larger than the DR—that is $\lfloor \mathsf{DR}(\mathcal{X}) \rfloor + 1$—quantifies how many *bits* are required to faithfully represent $\mathcal{X}$.

We also mention two different measures for describing the required precision.

[2] https://docs.dwavesys.com/docs/latest/c_qpu_ice.html (last accessed September 19, 2025)

Definition 4.3 (Coefficient Ratio [164])**.** Let $\mathcal{X} \subset \mathbb{R}$ be a finite set. The *coefficient ratio* (CR) of $\mathcal{X}$ is defined as

$$\mathsf{CR}(\mathcal{X}) = \frac{\hat{D}(\mathcal{X}, \{0\})}{\check{D}(\mathcal{X}, \{0\})} \, . \tag{4.3}$$

Definition 4.4 (Bit-Width [163])**.** Let $\mathcal{X} \subset \mathbb{Z}$ be a finite set. The *bit-width* (BW) of $\mathcal{X}$ is defined as

$$\mathsf{BW}(\mathcal{X}) = \left\lceil \log_2 \left(\hat{D}(\mathcal{X}, \{0\}) \right) \right\rceil + 1 \, . \tag{4.4}$$

For describing the DR, CR and BW of a QUBO matrix $\boldsymbol{Q}$, we use the shorthand notations $\mathsf{DR}(\boldsymbol{Q}) \equiv \mathsf{DR}(\mathrm{set}(\boldsymbol{Q}))$, $\mathsf{CR}(\boldsymbol{Q}) \equiv \mathsf{CR}(\mathrm{set}(\boldsymbol{Q}))$ and $\mathsf{BW}(\boldsymbol{Q}) \equiv \mathsf{BW}(\mathrm{set}(\boldsymbol{Q}))$, where $\mathrm{set}(\boldsymbol{Q}) \coloneqq \{Q_{ij} : i, j \in [n]\}$. Note that always $0 \in \mathrm{set}(\boldsymbol{Q})$, since $\boldsymbol{Q}$ is upper triangular, that is $Q_{ij} = 0$ for $i > j$. We want to give theoretical relationships between the three precision measures DR, CR and BW, but we note that the BW is only defined for integer entries. However, this does not pose a problem with respect to QUBO matrix entries.

Proposition 4.1. *Let $\boldsymbol{Q} \in \mathbb{R}^{n \times n}$. Then, there exists a $\boldsymbol{Q}' \in \mathbb{Q}^{n \times n}$, s.t. the QUBO energy landscapes of $E_{\boldsymbol{Q}}$ and $E_{\boldsymbol{Q}'}$ are equivalent, in the sense that*

$$E_{\boldsymbol{Q}}(\boldsymbol{z}) \leq E_{\boldsymbol{Q}}(\boldsymbol{z}') \Leftrightarrow E_{\boldsymbol{Q}'}(\boldsymbol{z}) \leq E_{\boldsymbol{Q}'}(\boldsymbol{z}') \; \forall \boldsymbol{z} \in \mathbb{B}^n \, . \tag{4.5}$$

Proof. For showing necessity, we first note that $\mathbb{Q}$ is dense in $\mathbb{R}$, that is

$$\forall q_1, q_2 \in \mathbb{Q} : \exists r \in \mathbb{R} : q_1 < r < q_2 \quad \text{and} \quad \forall r_1, r_2 \in \mathbb{R} : \exists q \in \mathbb{Q} : r_1 < q < r_2 \, .$$

We examine the smallest energy gap, i. e.,

$$E^{\Delta}_{\boldsymbol{Q}} \coloneqq \min_{\boldsymbol{z}, \boldsymbol{z}' \in \mathbb{B}^n, \boldsymbol{z} \neq \boldsymbol{z}'} \left| E_{\boldsymbol{Q}}(\boldsymbol{z}) - E_{\boldsymbol{Q}}(\boldsymbol{z}') \right| \in \mathbb{R} \, ,$$

which leads to

$$E_{\boldsymbol{Q}}(\boldsymbol{z}) \leq E_{\boldsymbol{Q}}(\boldsymbol{z}') \Leftrightarrow E_{\boldsymbol{Q}}(\boldsymbol{z}) \leq E_{\boldsymbol{Q}}(\boldsymbol{z}') - E^{\Delta}_{\boldsymbol{Q}} \quad \forall \boldsymbol{z}, \boldsymbol{z}' \in \mathbb{B}^n, \boldsymbol{z} \neq \boldsymbol{z}' \, . \tag{4.6}$$

Due to density, we can set

$$Q'_{ij} = \min_{q \in \mathbb{Q}} \left\{ q : Q_{ij} - \frac{E^{\Delta}_{\boldsymbol{Q}}}{2n^2} < q < Q_{ij} + \frac{E^{\Delta}_{\boldsymbol{Q}}}{2n^2} \right\} \, .$$

This leads to

$$E_{\boldsymbol{Q}'}(\boldsymbol{z}) = \sum_{i,j \in [n]} Q'_{ij} z_i z_j < \sum_{i,j \in [n]} \left(Q_{ij} + \frac{E^{\Delta}_{\boldsymbol{Q}}}{2n^2} \right) z_i z_j \leq E_{\boldsymbol{Q}}(\boldsymbol{z}) + \frac{E^{\Delta}_{\boldsymbol{Q}}}{2} \, ,$$

and with similar arguments we deduce a lower bound

$$E_{\boldsymbol{Q}}(\boldsymbol{z}) - \frac{E^{\Delta}_{\boldsymbol{Q}}}{2} < E_{\boldsymbol{Q}'}(\boldsymbol{z}) < E_{\boldsymbol{Q}}(\boldsymbol{z}) + \frac{E^{\Delta}_{\boldsymbol{Q}}}{2} \, . \tag{4.7}$$

Finally, we obtain

$$E_{\boldsymbol{Q}'}(\boldsymbol{z}) < E_{\boldsymbol{Q}}(\boldsymbol{z}) + \frac{E_{\boldsymbol{Q}}^{\Delta}}{2} \leq E_{\boldsymbol{Q}}(\boldsymbol{z}') - E_{\boldsymbol{Q}}^{\Delta} + \frac{E_{\boldsymbol{Q}}^{\Delta}}{2} < E_{\boldsymbol{Q}'}(\boldsymbol{z}') \quad \forall \boldsymbol{z}, \boldsymbol{z}' \in \mathbb{B}^n, \boldsymbol{z} \neq \boldsymbol{z}' \ ,$$

by using Equations (4.6) and (4.7). Sufficiency follows similarly. □

Proposition 4.2. *Let $\boldsymbol{Q} \in \mathbb{Q}^{n \times n}$. Then, there exists a $\boldsymbol{Q}' \in \mathbb{Z}^{n \times n}$, s.t. the QUBO energy landscapes of $E_{\boldsymbol{Q}}$ and $E_{\boldsymbol{Q}'}$ are equivalent, according to Equation (4.5).*

Proof. We note that the QUBO energy is scale equivariant, that is $E_{\alpha \boldsymbol{Q}} = \alpha E_{\boldsymbol{Q}}$, $\alpha \in \mathbb{R}$. Since $\boldsymbol{Q} \in \mathbb{Q}^{n \times n}$, every entry can be written as $Q_{ij} = \frac{p}{q}$, $p_{ij}, q_{ij} \in \mathbb{Z}$. Setting $\alpha \coloneqq \prod_{i,j \in [n]} q_{ij}$, we find that $\boldsymbol{Q}' \coloneqq \alpha \boldsymbol{Q} \in \mathbb{Z}^{n \times n}$. Due to scale equivariance, we obtain the claim. □

Combining Propositions 4.1 and 4.2, we deduce a very interesting insight.

Corollary 4.1. *Let $\boldsymbol{Q} \in \mathbb{R}^{n \times n}$. Then, there exists a $\boldsymbol{Q}' \in \mathbb{Z}^{n \times n}$, s.t. the QUBO energy landscapes of $E_{\boldsymbol{Q}}$ and $E_{\boldsymbol{Q}'}$ are equivalent, according to Equation (4.5).*

Corollary 4.1 tells us, that we can assume that our QUBO matrix $\boldsymbol{Q}$ as integer entries, without loss of generality. We can now compare the different precision measures.

Proposition 4.3. *Let $\boldsymbol{Q} \in \mathbb{Z}^{n \times n}$. Then*

$$\mathsf{BW}(\boldsymbol{Q}) - 2 - \log_2\left(\check{D}(\mathrm{set}(\boldsymbol{Q}), \{0\})\right) \leq \log_2\left(\mathsf{CR}(\boldsymbol{Q})\right) \leq \mathsf{DR}(\boldsymbol{Q}) \ .$$

Proof. The first inequality directly follows from definition. For the second one, not that

$$\hat{D}(\mathrm{set}(\boldsymbol{Q}), \{0\}) \leq \hat{D}(\mathrm{set}(\boldsymbol{Q})), \quad \check{D}(\mathrm{set}(\boldsymbol{Q}), \{0\}) \geq \check{D}(\mathrm{set}(\boldsymbol{Q})) \ ,$$

which leads to the desired result. □

The BW does not capture inter-weight distances, making it an inaccurate measure when scaling parameters to a specific range. Even though CR might be very small, the DR can still be large, but the reverse does not hold. That is to say, we understand DR as an accurate measure of representational complexity.

4.2.2 Precision Reduction While Preserving Optima

Every QUBO instance has at least one binary vector with minimum energy, which is the global optimum of the optimization problem. Scaling the QUBO matrix with a positive factor α does not change the optima, therefore $\mathcal{Z}^*(\boldsymbol{Q}) = \mathcal{Z}^*(\alpha \boldsymbol{Q})$, where we remember $\mathcal{Z}^*(\boldsymbol{Q})$ defined to be the set of all optimizers of $E_{\boldsymbol{Q}}$ (see Definition 2.7). Note that $\forall n \in \mathbb{N} : \forall \boldsymbol{Q} \in \mathcal{Q}_n : \mathcal{Z}^*(\boldsymbol{Q}) \neq \emptyset$, as in a non-empty finite set of real numbers a smallest element always exists.

Having a choice between multiple optima has several benefits. For one, when there is a variety of equally good solutions, we can choose one according to other criteria that are not encoded in the optimization problem (e. g., the solution with fewest 1-bits). Moreover, certain iterative optimization strategies may converge more quickly when there are multiple global optima scattered through the search

space, as the distance to the nearest optimum across all binary vectors decreases. However, when we are interested in just *any* optimal solution, we have to accept that, when modifying the QUBO matrix, some optima may get lost. To this end, we define the notion of *optimum inclusion* on QUBO instances.

Definition 4.5 (Optimum inclusion)**.** Let $\boldsymbol{Q}, \boldsymbol{Q}' \in \mathcal{Q}_n$. We say that $\boldsymbol{Q}$ *includes the optima of* $\boldsymbol{Q}'$, written as $\boldsymbol{Q} \trianglelefteq^* \boldsymbol{Q}'$, when the set of optima of the $\boldsymbol{Q}$-instance is a subset of the optima of the $\boldsymbol{Q}'$-instance:

$$\boldsymbol{Q} \trianglelefteq^* \boldsymbol{Q}' \;\Leftrightarrow\; \mathcal{Z}^*\left(\boldsymbol{Q}\right) \subseteq \mathcal{Z}^*\left(\boldsymbol{Q}'\right) \;.$$

The relation $\trianglelefteq^*$ induces a preorder on $\mathcal{Q}_n$. It is not anti-symmetric, as $\boldsymbol{Q} \trianglelefteq^* \boldsymbol{Q}'$ and $\boldsymbol{Q}' \trianglelefteq^* \boldsymbol{Q}$ does not imply $\boldsymbol{Q} = \boldsymbol{Q}'$, but it is reflexive and transitive. Informally, if $\boldsymbol{Q} \trianglelefteq^* \boldsymbol{Q}'$, we know that the QUBO instances with $\boldsymbol{Q}$ and $\boldsymbol{Q}'$ share at least one global optimum, and that the $\boldsymbol{Q}$-instance has no optima that the $\boldsymbol{Q}'$-instance does not have as well. Therefore, a binary vector that minimizes $E_{\boldsymbol{Q}}$ is guaranteed to also minimize $E_{\boldsymbol{Q}'}$.

We formulate the precision reduction problem as follows.

Definition 4.6. Let $\boldsymbol{Q} \in \mathcal{Q}_n$. To reduce the needed precision for representing $\boldsymbol{Q}$ while maintaining an optimizer, we define the objective as

$$\underset{\boldsymbol{A} \in \mathcal{Q}_n}{\arg\min} \quad \mathsf{DR}(\boldsymbol{Q} + \boldsymbol{A}) \tag{4.8a}$$

$$\text{s.t.} \quad \boldsymbol{Q} + \boldsymbol{A} \trianglelefteq^* \boldsymbol{Q} \;. \tag{4.8b}$$

Let us give a small example for clarification.

Example 3. Consider the following 2×2 matrices:

$$\boldsymbol{Q} = \begin{bmatrix} 0.8 & -1.5 \\ 0 & -1000 \end{bmatrix}, \quad \boldsymbol{Q}' = \begin{bmatrix} 0.8 & -1.5 \\ 0 & -2 \end{bmatrix} \;.$$

Observing the corresponding QUBO problems, we get

$$\underset{\boldsymbol{z} \in \mathbb{B}^2}{\arg\min}\, \boldsymbol{z}^\top \boldsymbol{Q} \boldsymbol{z} = \underset{\boldsymbol{z} \in \mathbb{B}^2}{\arg\min}\, \boldsymbol{z}^\top \boldsymbol{Q}' \boldsymbol{z} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \;,$$

that is, $\boldsymbol{Q}$ and $\boldsymbol{Q}'$ have the same optimizer, therefore $\boldsymbol{Q} \trianglelefteq^* \boldsymbol{Q}'$. Furthermore, it holds that

$$\boldsymbol{Q} + \boldsymbol{A} = \boldsymbol{Q}', \; \boldsymbol{A} \coloneqq \begin{bmatrix} 0 & 0 \\ 0 & 998 \end{bmatrix} \;.$$

When we compare the DR of $\boldsymbol{Q}$ and $\boldsymbol{Q}'$, we find that $\mathsf{DR}(\boldsymbol{Q}) \approx 10.29$, $\mathsf{DR}(\boldsymbol{Q}') \approx 2.49$.

Example 3 demonstrates that, in principle, it is possible to reduce the DR while preserving an optimizer of the QUBO problem. Interestingly, finding an optimal solution to Equation (4.8) is generally as hard as solving QUBO problem itself.

Proposition 4.4. *Let $\boldsymbol{Q} \in \mathcal{Q}_n$. Solving Equation (4.8) is NP-hard.*

Proof. We can find an optimal solution $\boldsymbol{A}^* = \left(\delta_{ij}(1 - 2z_i^*)\right)_{i,j=1}^n - \boldsymbol{Q}$ to Equation (4.8), when we already know an optimizer $\boldsymbol{z}^* \in \mathcal{Z}^*(\boldsymbol{Q})$. $\boldsymbol{Q} + \boldsymbol{A}^*$ is a diagonal matrix only consisting of the entries

$-1, 0, 1$ with a minimum $\mathrm{DR}(\boldsymbol{Q} + \boldsymbol{A}^*) = 1$. Nevertheless, solving an arbitrary QUBO problem with matrix $\boldsymbol{Q}$ is NP-hard, but the resulting optimum of Equation (4.8) is a diagonal matrix, for which the corresponding QUBO problem is solvable in linear time $\mathcal{O}(n)$. Solving Equation (4.8) is also NP-hard and thus as intractable to solve as the QUBO problem itself. □

4.2.3 Bounds for Preserving Optima

In general, reducing the DR leads to coarser energy gradations. E.g., a QUBO instance where parameters are encoded with only 2 bits can only have values in $\{-2, -1, 0, 1\}$, which may be insufficient to accurately preserve the value function and, consequently, the minimizing binary vectors. In this section we develop strategies to balance these competing objectives and reduce the DR while keeping (some of) the optimal vectors intact. To this end, we modify parameter values while trying to stay within bounds which guarantee that a minimal solution stays minimal, and a non-minimal solution does not become minimal.

To approach this problem, we update the elements of $\boldsymbol{Q}$ sequentially by assigning $Q_{kl} \mapsto Q_{kl} + w_{kl}$ for index pairs $k, l \in [n]$ with $k \leq l$. Deciding how to choose (i) the indices k and l, and (ii) the update value w_{kl} will be the focus of the following subsections. For now, let the indices $k \leq l$ be arbitrary but fixed.

Interval for Parameter Change Recall our definition of $\mathbb{B}^n_{I \leftarrow \boldsymbol{\zeta}}$ from Definition 2.3. Fixing one or more bits in a binary vector to constants induces subspaces of $\mathbb{B}^n$, one for each possible assignment of variables indexed by I, which is $2^{|I|}$ in total. Each subspace has its own set of minimizing binary vectors w.r.t. $E_{\boldsymbol{Q}}$.

Definition 4.7. Let $\boldsymbol{Q} \in \mathcal{Q}_n$, $I \subseteq [n]$ and $\boldsymbol{\zeta} \in \mathbb{B}^{|I|}$. The set of subspace optima for assignment $\boldsymbol{\zeta}$ is defined as

$$\mathcal{Z}^*_{I \leftarrow \boldsymbol{\zeta}}(\boldsymbol{Q}) \coloneqq \operatorname*{arg\,min}_{\boldsymbol{z} \in \mathbb{B}^n_{I \leftarrow \boldsymbol{\zeta}}} E_{\boldsymbol{Q}}(\boldsymbol{z}) \ .$$

Let $y^*_{ab} \coloneqq \min_{\boldsymbol{z} \in \mathcal{Z}^*_{kl \leftarrow ab}(\boldsymbol{Q})} E_{\boldsymbol{Q}}(\boldsymbol{z})$ denote the optimal value of the subspace created through the assignment $(a, b)^\top \in \mathbb{B}^2$. For conciseness, we write ab instead of $kl \leftarrow ab$ in the index, making k and l implicit from now on. Naturally, y^*_{ab} is just as hard to compute as solving QUBO itself. Therefore, we work with upper/lower bounds on the true values, which are much easier to compute. We use them to determine the update parameter w_{kl}.

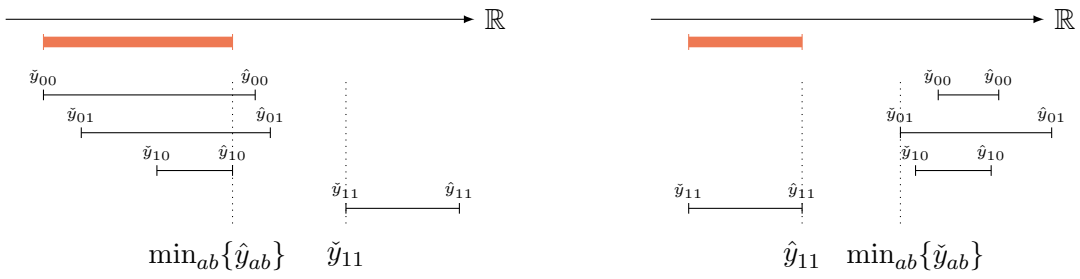
Definition 4.8. Let y^*_{ab} be defined as before. Denote upper and lower bounds on y^*_{ab} by $\check{y}_{ab}$ and $\hat{y}_{ab}$, i. e.,

$$\check{y}_{ab} \leq y^*_{ab} \leq \hat{y}_{ab}, \ \forall (a, b) \in \mathbb{B}^2 \ .$$

Further, we define

$$\begin{aligned} y^-_{kl} &\coloneqq \min\{0, \min\{\hat{y}_{00}, \hat{y}_{01}, \hat{y}_{10}\} - \check{y}_{11}\} \ , \\ y^+_{kl} &\coloneqq \max\{0, \min\{\check{y}_{00}, \check{y}_{01}, \check{y}_{10}\} - \hat{y}_{11}\} \ , \end{aligned}$$

if $k \neq l$. Otherwise, let $y^-_{kl} = \min\{0, \hat{y}_{00} - \check{y}_{11}\}$ and $y^+_{kl} = \max\{0, \check{y}_{00} - \hat{y}_{11}\}$ $(k = l)$.



(a) $\check{y}_{11} \geq \min\{\hat{y}_{00}, \hat{y}_{01}, \hat{y}_{10}\} \Rightarrow \boldsymbol{z}^* \notin \mathcal{Z}^*_{kl\leftarrow 11}(\boldsymbol{Q})$.

(b) $\hat{y}_{11} \leq \min\{\check{y}_{00}, \check{y}_{01}, \check{y}_{10}\} \Rightarrow \boldsymbol{z}^* \in \mathcal{Z}^*_{kl\leftarrow 11}(\boldsymbol{Q})$.

Figure 4.2: Illustration of Proposition 4.5: The orange bars indicate the interval the global optimum must fall into. When the lower bound for a subspace $\mathbb{B}^n_{kl\leftarrow ab}$ is greater than an upper bound of any other subspace, we can conclude that $\mathcal{Z}^*_{kl\leftarrow 11}(\boldsymbol{Q})$ does not contain an optimizer (a). On the other hand, when an the upper bound for a subspace $\mathbb{B}^n_{kl\leftarrow ab}$ is lower than the lower bounds of all other subspaces, we can conclude that an optimizer is in $\mathcal{Z}^*_{kl\leftarrow 11}(\boldsymbol{Q})$ (b). For the above example, we set $ab = 11$.

Theorem 4.1. *Let $\boldsymbol{Q} \in \mathcal{Q}_n$, $k, l \in [n]$ and consider updating the QUBO matrix parameter $Q_{kl} \mapsto Q_{kl} + w_{kl}$. The resulting QUBO matrix preserves an optimum if*

$$y^-_{kl} \leq w_{kl} \leq y^+_{kl} \,. \tag{4.9}$$

Proof. We assume $k \neq l$, since a similar argument holds for $k = l$. Note that one of the four subspaces' minimum energies $y^*_{00}, y^*_{01}, y^*_{10}, y^*_{11}$ corresponds to the global minimum y^* and changing Q_{kl} by w_{kl} affects only y^*_{11}. Assuming $\boldsymbol{z}^* \notin \mathcal{Z}^*_{kl\leftarrow 11}(\boldsymbol{Q})$, an optimum is preserved if

$$\begin{aligned} & y^*_{11} + w_{kl} \geq y^* \\ \Leftrightarrow\ & y^*_{11} + w_{kl} \geq \min\{y^*_{00}, y^*_{01}, y^*_{10}\} \\ \Leftarrow\ & \check{y}_{11} + w_{kl} \geq \min\{\hat{y}_{00}, \hat{y}_{01}, \hat{y}_{10}\} \,. \end{aligned} \tag{4.10}$$

Since $y^*_{11} > y^*$, w_{kl} can take any positive value. Combining this observation with Equation (4.10), we obtain lower bound

$$w_{kl} \geq \min\{0, \min\{\hat{y}_{00}, \hat{y}_{01}, \hat{y}_{10}\} - \check{y}_{11}\} = y^-_{kl} \,. \tag{4.11}$$

We can similarly deduce an upper bound for $\boldsymbol{z}^* \in \mathcal{Z}^*_{kl\leftarrow 11}(\boldsymbol{Q})$,

$$w_{kl} \leq \max\{0, \min\{\check{y}_{00}, \check{y}_{01}, \check{y}_{10}\} - \hat{y}_{11}\} = y^+_{kl} \,. \tag{4.12}$$

Our claim is obtained by combining Equations (4.11) and (4.12). □

Equation (4.9) uses bounds ($\check{y}_{ab}$ and $\hat{y}_{ab}$) on the true optima y^*_{ab} to give us an interval for w_{kl} if we want to preserve an optimum. The bounds in Equation (4.9) determine an interval $w_{kl} \in [y^-_{kl}, y^+_{kl}]$ for optimum preservation but can also be used for determining optimality of the subspaces.

Proposition 4.5. *The following implications hold:*

$$\check{y}_{ab} > \min\left(\{\hat{y}_{00}, \hat{y}_{01}, \hat{y}_{10}, \hat{y}_{11}\} \setminus \{\hat{y}_{ab}\}\right) \Rightarrow \boldsymbol{z}^* \notin \mathcal{Z}^*_{kl\leftarrow ab}(\boldsymbol{Q}), \tag{4.13}$$

$$\hat{y}_{ab} < \min\left(\{\check{y}_{00}, \check{y}_{01}, \check{y}_{10}, \check{y}_{11}\} \setminus \{\check{y}_{ab}\}\right) \Rightarrow \boldsymbol{z}^* \in \mathcal{Z}^*_{kl\leftarrow ab}(\boldsymbol{Q}) \,. \tag{4.14}$$

Proof. Assume Equation (4.13) holds, i. e.,

$$\begin{aligned} &\check{y}_{ab} > \min\left(\{\hat{y}_{00}, \hat{y}_{01}, \hat{y}_{10}, \hat{y}_{11}\} \setminus \{\hat{y}_{ab}\}\right) \\ \Rightarrow\ &y^*_{ab} > \min\left(\{y^*_{00}, y^*_{01}, y^*_{10}, y^*_{11}\} \setminus \{y^*_{ab}\}\right) \\ \Leftrightarrow\ &y^*_{ab} > y^* \ \Leftrightarrow\ \boldsymbol{z}^* \notin \mathcal{Z}^*_{kl\leftarrow ab}(\boldsymbol{Q}) \ . \end{aligned}$$

Analogously, the result in Equation (4.14) can be deduced. □

A visualization of Proposition 4.5 can be found in Figure 4.2. If we find the inequality in Equation (4.14) to be true for some two-bit assignment ab, the dimension of the QUBO search space can be reduced by fixing $z_k = a$ and $z_l = b$. Similar reduction techniques can be found in [168–170]. Knowing that $\boldsymbol{z}^* \notin \mathcal{Z}^*_{kl\leftarrow ab}(\boldsymbol{Q})$ we can get rid of the upper bound Equation (4.9) (cf. proof of Theorem 4.1).

Bounding the Optimal Value The questions remains how to obtain lower and upper bounds on y^*_{ab}. For this, we remark that the optimal energy value is always bounded from above by any energy value of the QUBO instance. Specifically, one can evaluate the vector consisting only of zeros.

Proposition 4.6. *Let $\boldsymbol{Q} \in \mathcal{Q}_n$, $k, l \in [n]$ and $(a, b) \in \mathbb{B}^2$. An upper bound for y^*_{ab} is given by*

$$y^*_{ab} \leq E_{\boldsymbol{Q}}(a\boldsymbol{e}_k + b\boldsymbol{e}_l) \ .$$

To obtain better upper bounds, we can invest more computational effort. For instance, one can perform a local or random search in the space $\mathbb{B}^n_{kl\leftarrow ab}$ or use more sophisticated methods, such as SA [62] or tabu search [7].

For obtaining a simple lower bound, we can sum over only the negative entries of $\boldsymbol{Q}$.

Proposition 4.7. *Let $\boldsymbol{Q} \in \mathcal{Q}_n$, $k, l \in [n]$ and $(a, b) \in \mathbb{B}^2$. Define $\boldsymbol{Q}^-$ as the matrix containing only the negative values of $\boldsymbol{Q}$, i. e., $Q^-_{ij} = \min\{0, Q_{ij}\}$. A lower bound for y^*_{ab} is given by*

$$y^*_{ab} \geq E_{\boldsymbol{Q}^-}(\mathbf{1} + (a-1)\boldsymbol{e}_k + (b-1)\boldsymbol{e}_l) \ .$$

The lower bound can be improved by exploiting more sophisticated techniques, e. g., through using roof duality [171, 172] or a continuous *Semi-Definite Programming* relaxtion [146]. Such methods come at cost of higher computational complexity, but still with polynomial runtimes.

4.3 Reducing the Dynamic Range

We have established intervals within which the parameters of a QUBO problem can be modified while an optimum is preserved. The question remains how to choose the values in a way that reduces the DR. In this section we show multiple approaches to achieve this goal.

Let $m := n^2$ be the number of entries of an n by n square matrix. For any $\boldsymbol{Q} \in \mathcal{Q}_n$ there is an ordering $\pi : [m] \to [n]^2$ of values in $\boldsymbol{Q}$ such that

$$q_i \leq q_{i+1},\ q_i \equiv Q_{\pi(i)},\ \forall i \in [m] \ .$$

Using this notation, $q_1 = \min \text{ set}(\boldsymbol{Q})$, $q_m = \max \text{ set}(\boldsymbol{Q})$, and further $\hat{D}(\boldsymbol{Q}) = q_m - q_1$ and $\check{D}(\boldsymbol{Q}) = \min\{q_{j+1} - q_j : j \in [m-1]\}$. Note that about half of all q_i are 0, as $\boldsymbol{Q}$ is upper triangular. An example is given in Figure 4.3.

Theorem 4.2. *Let $\boldsymbol{Q} \in \mathcal{Q}_n$, $k, l \in [n]$, $\pi(\ell) = (k, l)$, $Q_{kl} \neq 0$ and consider updating the QUBO matrix parameter $Q_{kl} \mapsto Q_{kl} + w_{kl}$. $\mathsf{DR}(\boldsymbol{Q}) \geq \mathsf{DR}(\boldsymbol{Q} + w_{kl}\boldsymbol{e}_k\boldsymbol{e}_l^\top)$, if the following two conditions hold:*

1. *w_{kl} is bounded:*

$$\underbrace{q_1 - q_\ell + \delta_{m\ell}\,(q_{m-1} - q_m) - D_\ell^*}_{=:d_\ell^-} \leq w_{kl} \leq \underbrace{q_m - q_\ell + \delta_{1\ell}\,(q_2 - q_1) + D_\ell^*}_{=:d_\ell^+}\,, \tag{4.15}$$

2. *w_{kl} does not decrease the minimal parameter distance:*

$$|q_\ell + w_{kl} - q_i| \geq \check{D}(\boldsymbol{Q}), \qquad \forall i \in [m] \setminus \{\ell\} \tag{4.16}$$

$$\vee \qquad q_\ell + w_{kl} = q_i, \qquad \exists i \in [m]\,. \tag{4.17}$$

Here, $\delta_{\cdot}$ is the Kronecker delta with $\delta_{uv} := 1$ if $u = v$, else 0, and D_ℓ^* is defined as

$$D_\ell^* := \hat{D}(\boldsymbol{Q})\left(\frac{\check{D}(\{q_u :\ u \in [m]\setminus\{\ell\}\})}{\check{D}(\boldsymbol{Q})} - 1\right)\,.$$

Proof. Assume $w_{kl} > 0$, the case $w_{kl} < 0$ follows analogously. Firstly, consider $q_\ell > q_1$. If

$$w_{kl} \leq q_m - q_\ell\,, \tag{4.18}$$

then $\hat{D}(\boldsymbol{Q})$ is not increased. Maintaining a distance of at least $\check{D}(\boldsymbol{Q})$ to all other QUBO parameters, i. e.,

$$|q_\ell + w_{kl} - q_i| \geq \check{D}(\boldsymbol{Q}),\ \forall i \in [m] \setminus \{\ell\}\,, \tag{4.19}$$

or "landing" on an already existing QUBO parameter, i. e.,

$$q_\ell + w_{kl} = q_i,\ \exists i \in [m]\,, \tag{4.20}$$

avoids a decrease of $\check{D}(\boldsymbol{Q})$ and thus the DR is not increased. On the other hand, $\hat{D}(\boldsymbol{Q})$ is increased if $w_{kl} > q_m - q_\ell$. Hence, to reduce the DR, $\check{D}(\boldsymbol{Q})$ has to increase. This can only happen if q_ℓ is unique and is part of the minimum distance, i. e., $\check{D}(\{q_u : u \in [m]\setminus\{\ell\}\}) > \check{D}(\boldsymbol{Q})$. We obtain a bound on w_{kl}

by

$$\begin{aligned}
&\mathsf{DR}\left(\boldsymbol{Q}\right) \geq \mathsf{DR}\left(\boldsymbol{Q} + w_{kl}\boldsymbol{e}_k\boldsymbol{e}_l^\top\right) \\
\Leftrightarrow\ &\frac{\hat{D}(\boldsymbol{Q})}{\check{D}(\boldsymbol{Q})} \geq \frac{\hat{D}(\boldsymbol{Q}) + w_{kl} - (q_m - q_\ell)}{\min\{\check{D}(\{q_u :\ u \in [m]\backslash\{\ell\}\}), \check{D}(\boldsymbol{Q}) + w_{kl}\}} \\
&\qquad \geq \frac{\hat{D}(\boldsymbol{Q}) + q_\ell + w_{kl} - q_m}{\check{D}(\{q_u :\ u \in [m]\backslash\{\ell\}\})} \\
&\Leftrightarrow w_{kl} \leq q_m - q_\ell + \hat{D}(\boldsymbol{Q})\left(\frac{\check{D}(\{q_u :\ u \in [m]\backslash\{\ell\}\})}{\check{D}(\boldsymbol{Q})} - 1\right) .
\end{aligned} \tag{4.21}$$

Secondly, consider $\ell = 1$ and assume q_1 is unique ($q_2 - q_1 > 0$), otherwise we can deduce bounds Equations (4.18) to (4.20). If $w_{kl} > q_2 - q_1$, q_2 is the new minimal value instead of $q_1 + w_{kl}$. Thus, we can add the difference $q_2 - q_1$ to the bound in Equation (4.18)

$$w_{kl} \leq (q_m - q_1) + (q_2 - q_1) = q_m - 2q_1 + q_2 .$$

If q_1 is also part of the unique minimum distance, we add $q_2 - q_1$ to the bound in Equation (4.21)

$$w_{kl} \leq q_m - q_1 + q_2 - q_1 + \hat{D}(\boldsymbol{Q})\left(\frac{\check{D}_{\text{set}}(\{q_u :\ u \in [m]\backslash\{\ell\}\})}{\check{D}(\boldsymbol{Q})} - 1\right) . \tag{4.22}$$

Similar bounds can be obtained for a negative change $w_{kl} < 0$. □

Theorem 4.2 provides loose bounds on the QUBO parameter changes. We discuss several heuristic approaches of how to choose w_{kl} within these bounds in the upcoming subsections.

4.3.1 Greedy Strategy

The first heuristic is a greedy strategy which we denote by G. The QUBO parameter Q_{kl} is increased if $q_\ell < 0$ and decreased otherwise, where $\pi(\ell) = (k, l)$. For increasing (decreasing) Q_{kl} we choose w_{kl}^{G} maximally (minimally), i. e., $w_{kl}^{\mathsf{G}} = d_\ell^+$ ($w_{kl}^{\mathsf{G}} = d_\ell^-$). If the updated QUBO parameter lays too close to some other parameter, i. e.,

$$\left|q_\ell + d_\ell^\pm - q_i\right| < \check{D}(\boldsymbol{Q}),\ \exists i \in [m] \setminus \{\ell\} ,$$

we set it equal to the next smaller (larger) QUBO parameter, that is,

$$q_\ell + w_{kl}^{\mathsf{G}} = q_i,\ q_j \leq q_\ell + d^+,\ \forall j \leq i\ \left(q_j \geq q_\ell + d^-,\ \forall j \geq i\right) . \tag{4.23}$$

Again, recall that there is always a $q_u = 0$ for some $u \in [m]$, and thus we may set parameters to 0. For certain target platforms, such as quantum annealers, this is particularly beneficial, as setting a parameter to 0 allows to discard the coupling between the qubits indexed by k and l, which saves hardware resources. As an alternative version to Equation (4.23), we choose $w_{kl}^{\mathsf{G}_0}$ such that

$$q_\ell + w_{kl}^{\mathsf{G}_0} = 0,\ 0 \leq q_\ell + d^+,\ \left(0 \geq q_\ell + d^-\right) .$$

We henceforth call this alternative version G_0.

4.3.2 Maintaining the Parameter Ordering

With the preceding methods, we allowed parameters to cross over each other, changing their ordering. However, another heuristic approach is to maintain the ordering of the elements in $\mathrm{set}(\boldsymbol{Q})$, which should intuitively help to preserve the optimum. This heuristic is denoted by M and we define bounds on a certain QUBO parameter q_ℓ

$$\hat{q}_\ell^+ \coloneqq \min\{q_t : \ q_t > q_\ell, t \in [m]\} \ , \tag{4.24a}$$

$$\check{q}_\ell^+ \coloneqq \max\{q_t : \ q_t \leq q_\ell, t \in [m] \setminus \{\ell\}\} \ , \tag{4.24b}$$

$$\hat{q}_\ell^- \coloneqq \min\{q_t : \ q_t \geq q_\ell, t \in [m] \setminus \{\ell\}\} \ , \tag{4.24c}$$

$$\check{q}_\ell^- \coloneqq \max\{q_t : \ q_t < q_\ell, t \in [m]\} \ . \tag{4.24d}$$

If all entries of $\boldsymbol{Q}$ are unique, then $\hat{q}_\ell^+ = \hat{q}_\ell^-$ and $\check{q}_\ell^+ = \check{q}_\ell^-$, so only if duplicate values exist, these bounds on q_ℓ differ. An example clarifying these bounds is given in Example 4. The idea is now that q_ℓ is changed in such a way that it lies exactly in the middle between $\check{q}_\ell^\pm$ and $\hat{q}_\ell^\pm$. For $q_1 < q_\ell < q_m$ we increase q_ℓ if $q_\ell - \check{q}_\ell^- < \hat{q}_\ell^+ - q_\ell$ and decrease otherwise. The weight q_ℓ is thus changed by

$$w_{kl}^{\mathsf{M}} = \begin{cases} \frac{\hat{q}_\ell^+ - \check{q}_\ell^+}{2} - \min\{\hat{q}_\ell^+ - q_\ell, q_\ell - \check{q}_\ell^+\}, & \text{if } q_\ell - \check{q}_\ell^- < \hat{q}_\ell^+ - q_\ell, \\ \frac{\check{q}_\ell^- - \hat{q}_\ell^-}{2} + \min\{\hat{q}_\ell^- - q_\ell, q_\ell - \check{q}_\ell^-\}, & \text{else.} \end{cases}$$

For the edge case $\ell = m$, w_{kl}^{M} is given by

$$w_{kl}^{\mathsf{M}} = \begin{cases} \check{q}_m^- - q_m + \check{D}(\boldsymbol{Q}), & \text{if } \check{D}_{\mathrm{set}}(\{q_u : \ u \in [m]\backslash\{m\}\}) = \check{D}(\boldsymbol{Q}), \\ \check{D}_{\mathrm{set}}(\{q_u : \ u \in [m]\backslash\{m\}\}) - \check{D}(\boldsymbol{Q}), & \text{else.} \end{cases}$$

Similarly, for $\ell = 1$

$$w_{kl}^{\mathsf{M}} = \begin{cases} \hat{q}_1^+ - q_1 - \check{D}(\boldsymbol{Q}), & \text{if } \check{D}_{\mathrm{set}}(\{q_u : \ u \in [m]\backslash\{1\}\}) = \check{D}(\boldsymbol{Q}), \\ \check{D}(\boldsymbol{Q}) - \check{D}_{\mathrm{set}}(\{q_u : \ u \in [m]\backslash\{1\}\}), & \text{else.} \end{cases}$$

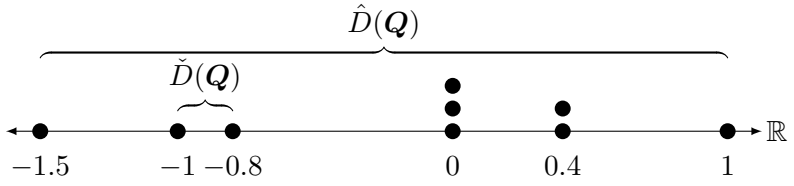
Having the heuristic change w_{kl}^h at hand, we can determine the final change w_{kl} as

$$w_{kl} = \min\{\max\{w_{kl}^h, y_{kl}^-\}, y_{kl}^+\} \ , \tag{4.25}$$

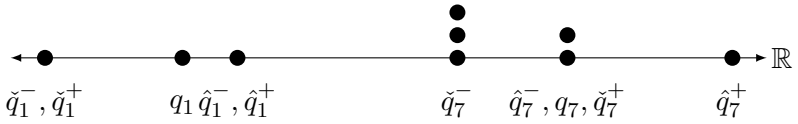
which ensures that $w_{kl} \in [y_{kl}^-, y_{kl}^+]$, such that the optimum is preserved.

Example 4. Consider the following exemplary QUBO matrix with $n = 3$:

$$\boldsymbol{Q} \coloneqq \begin{bmatrix} -1 & 0.4 & 1 \\ 0 & 0.4 & -0.8 \\ 0 & 0 & -1.5 \end{bmatrix} , \tag{4.26}$$



(a) We can read off $\hat{D}(\boldsymbol{Q}) = 2.5$ and $\check{D}(\boldsymbol{Q}) = 0.2$.



(b) Bounds (Equations (4.24a) and (4.24d)) on QUBO parameters $q_1 = -1$ and $q_7 = 0.4$.

Figure 4.3: Sorted QUBO parameters of the matrix given in Equation (4.26). Duplicates are indicated as vertically stacked points.

with $\mathrm{DR}(\boldsymbol{Q}) = \log_2(2.5/0.2) = 3.64$. The parameter ordering $q_1, \ldots, q_9$ is given by

$$(q_1, \ldots, q_9) = (-1.5, -1, -0.8, 0, 0, 0, 0.4, 0.4, 1) \ ,$$

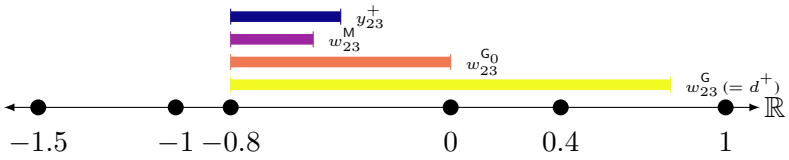
and is visualized in Figure 4.3. The bounds from Equations (4.24a) and (4.24d) for two specific parameters can be found in Figure 4.3(b). We want to increase the value $Q_{23} \equiv q_3 = -0.8$, because this would decrease $\check{D}(\boldsymbol{Q})$ and thus decrease the DR. We fix $k = 2$, $l = 3$ and find that $\boldsymbol{z}^* = (0, 1, 1)^\top$. For maintaining the optimum $\boldsymbol{z}^*$ when changing Q_{23}, we need to obey Theorem 4.1. Computing accurate bounds, i. e., the exact values, we obtain

$$\check{y}_{00} = 0, \ \check{y}_{01} = -1.5, \ \check{y}_{10} = 0.4, \ \hat{y}_{11} = -1.9,$$

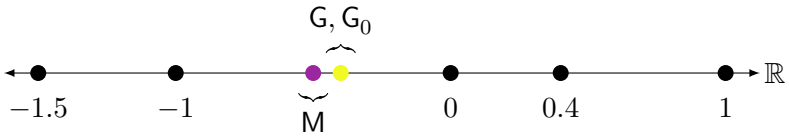
and thus $y_{kl}^+ = \min\{0, -1.5, 0.4\} - 1.9 = 0.4$. In words, we can maximally increase Q_{23} by 0.4 to maintain the optimum state, which is shown in Figure 4.4(a). For decreasing the DR we examine the three heuristics M, G and G_0. The values $w_{kl}^{\mathsf{M}} = 0.3$, $w_{kl}^{\mathsf{G}} = 1.6$ and $w_{kl}^{\mathsf{G}_0} = 0.8$ are also indicated in Figure 4.4(a). We observe that M changes Q_{23} to lie in the middle between its neighbors, G maximally increases Q_{23} to maintain the DR, while G_0 sets Q_{23} to 0. In Figure 4.4(b) the final changes are shown, using the three heuristics. Following Equation (4.25), the changes are given by 0.3 for M and 0.4 for G and G_0, respectively. Both result in a doubling of $\check{D}(\boldsymbol{Q})$ leading to a new DR decreased by one bit, i.e., 2.64.

4.3.3 Markov Decision Process Formulation

Since the changes of the QUBO parameters are carried out in a successive fashion, it remains to decide which $k, l \in [n], k \leq l$ to pick next. A very simple approach is to pick a random pair of indices, or iterate over all index pairs in sequence. Using this method, many—or, with growing n, most—iterations will not lead to a DR improvement, as only a few different parameters directly determine the DR, namely those closest and furthest apart (c.f. Figure 4.3(a)). Conversely, changing such an "inactive" parameter can never lead to a decrease in DR, only to an increase. This realization leads to a better strategy, which is choosing only among those index pairs whose parameters determine DR. In our experiments,



(a) Parameter value intervals: Maximum increase $y_{23}^{+} = 0.4$ (blue) of QUBO parameter Q_{23} such that the optimum is preserved, along with heuristic interval limits $w_{23}^{\mathsf{M}} = 0.3$ (purple), $w_{23}^{\mathsf{G}} = 1.6$ (orange) and $w_{23}^{\mathsf{G}_0} = 0.8$ (yellow).



(b) New parameters after changing $Q_{23} = -0.8$ w.r.t. preserving the optimum: $\min\{y_{23}^{+}, w_{23}^{\mathsf{M}}\} = 0.3$ for M (purple), and $\min\{y_{23}^{+}, w_{23}^{\mathsf{G}}\} = 0.4$ (yellow) for G (and G_0 analogously). In both cases, the DR decreases by one bit, since $\check{D}(\boldsymbol{Q})$ is doubled.

Figure 4.4: Change of QUBO parameter Q_{23}.

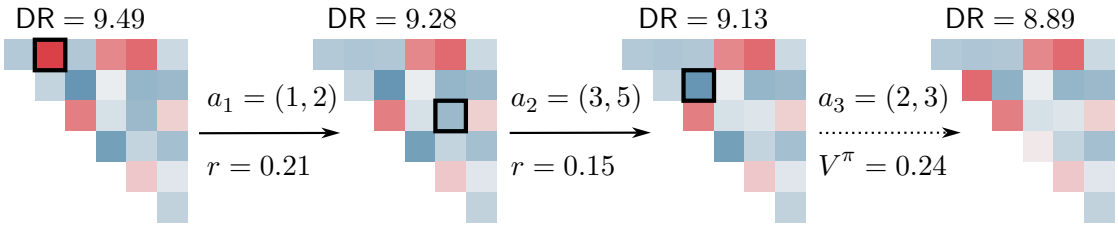


Figure 4.5: Illustration of the MDP described in Section 4.3.3. Every step t, we choose an action a_t in form of an index pair and update our state s_t to s_{t+1} to obtain a matrix with a smaller DR. The goal is to maximize the value function V^{π}.

we compute their respective update values a and greedily choose the one that leads to maximal DR reduction, breaking ties randomly.

However, this greedy myopic approach can easily end up in a local optimum. We present a more sophisticated long-sighted approach utilizing MDPs combined with B&B.

Recall the definition of an MDP from Section 2.3.3. We consider the state space as $\mathcal{S} = \mathcal{Q}_n$ and the action space $\mathcal{A} \subset [n] \times [n]$. The state transition function is given by $f(s, a) = f(\boldsymbol{Q}, (k, l)) = \boldsymbol{Q} + h(\boldsymbol{Q}, (k, l))\boldsymbol{e}_k\boldsymbol{e}_l^{\top}$, where $\boldsymbol{e}_k$, $\boldsymbol{e}_l$ are the standard basis vectors with zeros everywhere except at index k and l, respectively. $h : \mathcal{Q}_n \times [n] \times [n] \to \mathbb{R}$, is a function for determining the parameter update. For example, h can be chosen as one of presented heuristics in Section 4.3. We define the reward as the change of DR, $r(s, a) \coloneqq \mathsf{DR}(s) - \mathsf{DR}(f(s, a))$. The four-tuple $(\mathcal{S}, \mathcal{A}, f, r)$ defines an MDP. Now assume that we want to change T QUBO matrix entries such that the accumulated reward is maximized. Formally, the goal is to find a policy $\pi^* : \mathcal{S} \to \mathcal{A}$, s.t.,

$$\pi^* = \arg\max_{\pi:\mathcal{S}\to\mathcal{A}} V^{\pi}(s_0) = \arg\max_{\pi:\mathcal{S}\to\mathcal{A}} \sum_{t=0}^{T-1} r\left(s_t, \pi(s_t)\right) \tag{4.27a}$$

$$= \arg\min_{\pi:\mathcal{S}\to\mathcal{A}} \mathsf{DR}\left(f_T(\boldsymbol{Q}, \pi)\right) \ , \tag{4.27b}$$

where the equality follows through a telescopic sum and the t-time state transition f_t following policy π is defined as $f_t(s_0, \pi) \coloneqq f\left(s_t, \pi(s_t)\right) = s_{t+1}$, $s_0 \coloneqq \boldsymbol{Q}$. Our MDP is illustrated in Figure 4.5.

Observing that the transition is a simple matrix addition, we can write $f_T(\boldsymbol{Q}, \pi) = \boldsymbol{Q} + \boldsymbol{A}'$, where $\boldsymbol{Q} + \boldsymbol{A}' \boxplus \boldsymbol{Q}$. Thus, the optimization objective of the decision process in Equation (4.27) is a more restricted version of the problem in Equation (4.8). That is, we do not optimize over the set of all optimum inclusive matrices, but over the subset of matrices which can be created with any policy following our MDP framework. The cumulative sum in Equation (4.27a) is also called the value function V^{π} for a policy π. Using the recursive *Bellman equation*

$$V^{\pi^*}(s_t) = \max_{a \in \mathcal{A}} \left[r(s_t, a) + V^{\pi^*}\left(f(s_t, a)\right) \right] , \tag{4.28}$$

we can find an optimal policy in Equation (4.27) with dynamic programming (DP), using a shortest path-type method. In our case, we have no knowledge about the final state $f_T(\boldsymbol{Q}, \pi)$ and thus the search space is exponentially large (see Figure 4.6)

$$\sum_{t=0}^{T} (n^2)^t = \frac{(n^2)^{T+1} - 1}{n^2 - 1} \in \mathcal{O}(n^{2(T+1)}) . \tag{4.29}$$

Choosing the number of iterations T logarithmic in the problem dimension, i.e., $T = \log_2(n^2) = 2\log_2(n)$, results in the sub-exponential state space size of

$$\mathcal{O}\left(n^{4\log_2(n)+2}\right) = \mathcal{O}\left(n^2 2^{\left(4\log_2^2(n)\right)}\right) \subseteq o(2^n) .$$

Thus, we have an asymptotically slower growth than the exponentially large state space size 2^n of the original QUBO problem.

However, in practice, super-polynomial runtimes are often not tractable, especially for large n. Due to this fact, Equation (4.28) is typically solved with approximate DP methods such as Monte Carlo Tree Search (MCTS), PR or reinforcement learning [87, 94]. We present a B&B algorithm utilizing PR to reduce the complexity of solving Equation (4.27).

4.3.4 Branch and Bound

Following all paths of possible QUBO matrix updates is intractable. We combine PR with the B&B paradigm to obtain a trade-off between computational complexity and solution quality—solution paths which cannot lead to an optimum are pruned, based on bounds on the best found solution. The algorithm is given in Figure 4.6: the search space is expanded (branch) and every state is checked whether it can be pruned (bound). This is done until the final horizon T is reached and the state with the minimum DR is returned.

Branch In the *branch*-step, the search space is expanded from the current considered state. The question arises how to decide which indices to consider in the current iteration, i.e., which entries of QUBO matrix should be changed. The obvious method is to use all $n(n+1)/2$ upper triangular indices of the whole matrix, which we will further indicate by ALL. With large n, this expansion gets very large and thus we also consider a different method in our experiments. This method is based on the

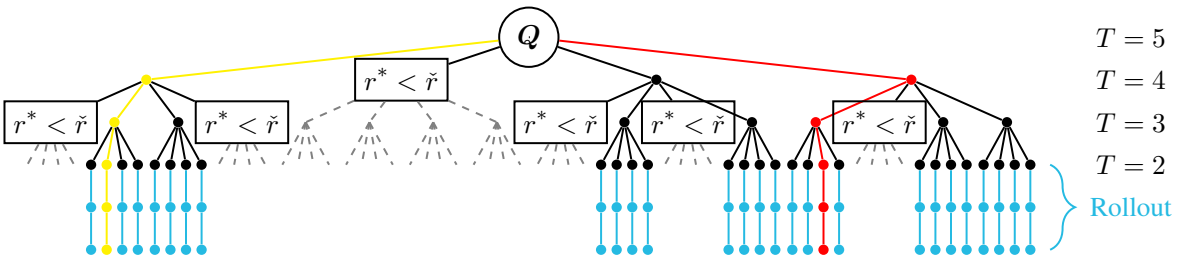


Figure 4.6: Exemplary depiction of the search space when applying our B&B algorithm (Section 4.3.4) to some QUBO matrix $\boldsymbol{Q}$. In every step, we expand our search space (from top to bottom, Section 4.3.4) and check whether a branch can be pruned ($r^* < \check{r}$, Section 4.3.4). The small filled circles indicate the visited states of our algorithm and the pruned parts are depicted as gray dashed lines. The fraction of pruned states is $40/85 = 0.47$ (without rollout). Since the search space size grows exponentially with the horizon $T = 5$, we execute a PR (Section 4.3.4) for $\tilde{T} = 2$ steps. From here on, a base policy is followed without expanding further, which is depicted in blue. The red path indicates the optimal solution and the yellow path shows the base policy.

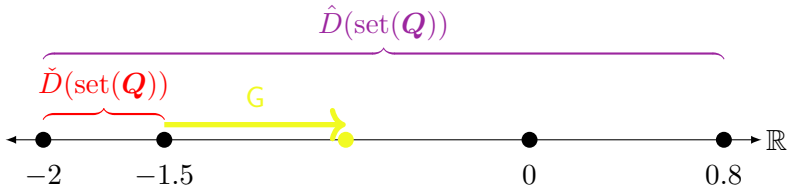
observation, that only four entries of the QUBO matrix affect the DR when changing a single weight. Namely the smallest/largest weight and the weights which are closest to each other, which will be denoted by IMPACT. This drastically reduces the search space search size and the performance to ALL is compared in Section 4.4.

Policy Rollout Having an index pair (k, l) at hand, a new state is created from the current state $\boldsymbol{Q}$ by following the transition function f. Instead of solving the problem in Equation (4.27) exactly and expanding the search space for T steps, we also consider a PR approach. It describes the concept of following a given a base policy $\dot{\pi}$ for a number of steps. For a given rollout depth $\tilde{T} \leq T$, we denote the policy which optimizes its path for $\tilde{T}$ steps and then follows $\dot{\pi}$ for $T - \tilde{T}$ steps as $\bar{\pi}_{\tilde{T}}$. We use a greedy policy $\dot{\pi}(\boldsymbol{Q}) \coloneqq \arg\min_{a \in \mathcal{A}} \mathsf{DR}\left(f(\boldsymbol{Q}, a)\right)$ which myopically optimizes the DR when taking a single step. Since the solution quality is monotonically increasing with $\tilde{T}$, we obtain a trade-off between the size of the state space and the performance of our algorithm. Using PR is motivated by the well known *rollout selection policy*. At time t, the optimal future reward $V^{\pi^*}\left(f(s_t, a)\right)$ is approximated with the reward $V^{\dot{\pi}}\left(f(s_t, a)\right)$ following $\dot{\pi}$. The Bellman equation Equation (4.28) is modified to

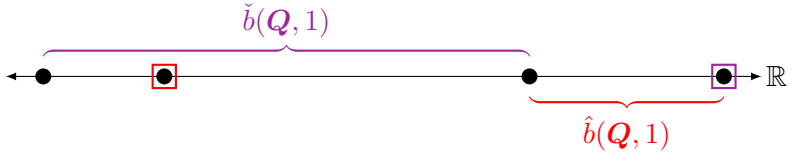
$$V^{\tilde{\pi}}(s_t) = \max_{a \in \mathcal{A}} \left[r(s_t, a) + V^{\dot{\pi}}\left(f(s_t, a)\right) \right] . \tag{4.30}$$

The resulting rollout selection policy $\tilde{\pi}$ is at least equal and typically better than the base policy $\dot{\pi}$. It is renowned for its simplicity and strong performance, largely due to its close relationship with the fundamental dynamic programming algorithm of policy iteration. Equation (4.30) represents the optimal one-step look-ahead policy, when subsequently following a base policy. This principle can be generalized to $\tilde{T}$-step look-ahead rollouts, $\tilde{T} \leq T$, where the solution quality increases with increasing rollout horizon $\tilde{T}$. The exact solution for Equation (4.27) is obtained if $\tilde{T} = T$. Thus, PR can be seamlessly integrated into our B&B algorithm. An experimental comparison between $\bar{\pi}_{\tilde{T}}$ and $\tilde{\pi}$ with using the aforementioned variants can be found in Section 4.4.

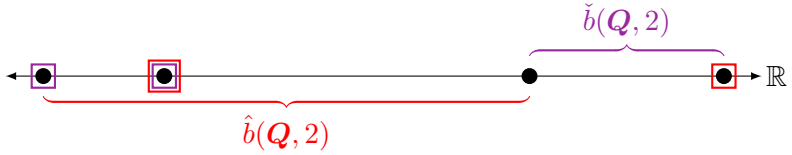
Bound We want to prune states, which cannot lead to the optimal solution. Deciding whether a state can be pruned, is dependent on bounds of the reachable best solution from that given state. Given the



(a) We can read off $\hat{D}(\mathrm{set}(\boldsymbol{Q})) = 2.8$ (purple) and $\check{D}(\mathrm{set}(\boldsymbol{Q})) = 0.5$ (red). Change of parameter Q_{01} using a heuristic $h(\boldsymbol{Q}, 0, 1) = 0.7$ (yellow). The sorted parameters are given by $q_1 = -2$, $q_2 = -1.5$, $q_3 = 0$ and $q_4 = 0.8$.



(b) Bounds when a single QUBO parameter is changed to 0: A lower bound (top, purple) is given by $\hat{D}(\mathrm{set}(f_1(\boldsymbol{Q}, \pi^*))) \geq \check{b}(\boldsymbol{Q}, 1) = 2$ and an upper bound (bottom, red) by $\check{D}(\mathrm{set}(f_1(\boldsymbol{Q}, \pi^*))) \leq \hat{b}(\boldsymbol{Q}, 1) = 0.8$. The changed parameters are indicated with rectangular boxes.



(c) Bounds when two QUBO parameters are changed, namely $\check{b}(\boldsymbol{Q}, 2) = 0.8$ and $\hat{b}(\boldsymbol{Q}, 2) = 2$. For details, see Figure 4.7(b).

Figure 4.7: Sorted QUBO matrix entries given in Example 3. Heuristic h is depicted in Figure 4.7(a) and the methods for finding a lower bound on the DR are given in Figures 4.7(b) and 4.7(c).

current best final DR r^*, we can prune a state $\boldsymbol{Q}$ if it is smaller than a lower bound $\check{r}(\boldsymbol{Q}, T)$ on the best reachable solution $\mathsf{DR}(f_T(\boldsymbol{Q}, \pi^*))$, i.e., if $r^* \leq \check{r}(\boldsymbol{Q}, T) \leq \mathsf{DR}(f_T(\boldsymbol{Q}, \pi^*))$. Pruning states, we do not have to expand the search further and can drastically reduce the computation time. We find a lower bound on $\mathsf{DR}(f_T(\boldsymbol{Q}, \pi^*))$ with a lower/upper bound $\check{b}(\boldsymbol{Q}, T)/\hat{b}(\boldsymbol{Q}, T)$ on the numerator/denominator in Equation (4.2)

$$\mathsf{DR}(f_T(\boldsymbol{Q}, \pi^*)) \geq \log_2 \left(\frac{\check{b}(\boldsymbol{Q}, T)}{\hat{b}(\boldsymbol{Q}, T)} \right) =: \check{r}(\boldsymbol{Q}, T) \ .$$

Let $m := n^2$ be the number of entries of an $n \times n$ matrix. For any $\boldsymbol{Q} \in \mathcal{Q}_n$, there is an ordering (bijective map) $\sigma : [m] \to [n] \times [n]$ of entries such that $q_\ell \leq q_{\ell+1},\ q_\ell \equiv Q_{\sigma(\ell)},\ \forall \ell \in [m]$. With this notation, $\hat{D}(\mathrm{set}(\boldsymbol{Q})) = q_m - q_1$ and $\exists j \in [m-1] : \check{D}(\mathrm{set}(\boldsymbol{Q})) = q_{j+1} - q_j$. A visualization for an ordering of Example 3 is shown in Figure 4.7(a).

Lower Bound on Maximum Distance For finding a lower bound on $\mathsf{DR}(f_T(\boldsymbol{Q}, \pi^*))$, we optimistically assume that we can set all parameters to 0 while maintaining an optimizer of $\boldsymbol{Q}$. Since 0 is always considered in the computation of the DR, this corresponds to an optimal strategy of changing the parameters, because the DR cannot increase. Changing a single parameter, the numerator $\hat{D}(\mathrm{set}(\boldsymbol{Q}))$ in Equation (4.2) is maximally reduced if we set q_1/q_m larger/smaller than q_2/q_{m-1}. The maximum possible reduction is equal to $\min\{q_2 - q_1, q_m - q_{m-1}\}$. Iterating this process for T times, we end up with

Algorithm 2 LOWERBOUND

Input: $\boldsymbol{Q}, T$
Output: Lower bound $\check{r} \leq \mathsf{DR}(\mathrm{set}(\boldsymbol{Q}_T^{\pi^*}))$
1: $\check{b} \leftarrow \min\{q_{m-T+i} - q_{i+1} : 0 \leq i \leq T\}$
2: Compute σ, s.t., $q_\ell \leq q_{\ell+1},\ q_\ell \equiv Q_{\sigma(\ell)}$ ▷ Sort weights
3: $\bar{\mathcal{D}} \leftarrow \{d_i : i \in [m-1]\},\ d_i = q_{i+1} - q_i$
4: Compute ρ, s.t., $d_{\rho(\ell)} \leq d_{\rho(\ell+1)}$ ▷ Sort distances
5: **for** $t = 1$ to T **do**
6: $\quad \mathcal{I} \leftarrow \{\rho(1), \rho(1)+1\}$
7: $\quad i_* = \arg\min_{i \in \mathcal{I}}\ d_i$
8: $\quad$ **if** $i_* = \rho(1)$ **then**
9: $\quad\quad i_* \leftarrow i_* - 1$
10: $\quad$ **end if**
11: $\quad d_{i_*} \leftarrow d_{i_*} + d_{\rho(1)}$ ▷ Update distance
12: $\quad \bar{\mathcal{D}} \leftarrow \bar{\mathcal{D}} \setminus \{d_{\rho(1)}\}$
13: $\quad$ Recompute ρ, s.t., $d_{\rho(\ell)} \leq d_{\rho(\ell+1)}$
14: **end for**
15: $\hat{b} \leftarrow d_{\rho(1)}$
16: $\check{r} \leftarrow \check{b}/\hat{b}$

$\check{b}(\boldsymbol{Q}, T) := \min\{q_{m-T+i} - q_{i+1} : 0 \leq i \leq T\}$. An illustration for Example 3 is found in Figures 4.7(b) and 4.7(c).

Upper Bound on Minimum Distance Obtaining a lower bound is a little more tricky and an iterative procedure is given in Algorithm 2. Since we are concerned with the smallest distance between two QUBO parameters, we consider the set of distances between "neighboring" parameters $\bar{\mathcal{D}}(\mathrm{set}(\boldsymbol{Q})) := \{q_{i+1} - q_i : i \in [m-1]\} = \{d_i : i \in [m-1]\}$. We iteratively set QUBO weights to 0, which are part of the minimum distance, maximizing the minimum distance. Define an ordering $\rho : [m-1] \to [m-1]$, s.t., $d_{\rho(i)} \leq d_{\rho(i+1)}$. It then holds that $\check{D}(\mathrm{set}(\boldsymbol{Q})) = d_{\rho(1)}$. $\check{D}(\mathrm{set}(\boldsymbol{Q}))$ is maximally reduced if we change $q_{\rho(1)+1}$ or $q_{\rho(1)}$, s.t., $d_{\rho(1)}$ is not the smallest distance anymore. We change the weight with the smaller corresponding distance (Lines 6 and 7, Algorithm 2). If $q_{\rho(1)}$ is changed, $d_{\rho(1)-1}$ is updated to $d_{\rho(1)-1} + d_{\rho(1)} = q_{\rho(1)+1} - q_{\rho(1)-1}$ and if $q_{\rho(1)+1}$ is changed, $d_{\rho(1)+1}$ is updated to $d_{\rho(1)+1} + d_{\rho(1)} = q_{\rho(1)+2} - q_{\rho(1)}$ (Algorithm 2, Algorithm 2). No update is required if $\rho(1) = 1$ or $\rho(1)+1 = m$. The new smallest distance either equals the second smallest distance $d_{\rho(2)}$ or one of the two newly updated ones. The smallest distance $d_{\rho(1)}$ is removed (Algorithm 2, Algorithm 2) from $\bar{\mathcal{D}}$ and the ordering ρ is updated with the updated distances (Algorithm 2, Algorithm 2). In Figures 4.7(b) and 4.7(c), this is illustrated for Example 3.

Computational Complexity Analysis Even though we are able to prune a large amount of the search space, it would be beneficial for the bounds to be computable efficiently. It turns out that the bounds can be dynamically computed in $\mathcal{O}(Tn^2)$. Through the use of memoization, this can be efficiently combined with PR.

For the transition $f(\boldsymbol{Q}, (k,l)) = \boldsymbol{Q} + h(\boldsymbol{Q}, (k,l))\boldsymbol{e}_k\boldsymbol{e}_l^\top$ we need to compute the value $h(\boldsymbol{Q}, (k,l))$

which is implicitly dependent on y^-_{kl} and y^+_{kl}. For computing a lower bound $\check{y}$, we use the roof dual technique [171]. A flow network is build with $\mathcal{O}(n)$ nodes, and the lower bound is given by the maximum flow value, which is computable in $\mathcal{O}(n^3)$. Exploiting the top-down nature of our B&B approach, we can dynamically update the flow network and recompute the maximum flow in $\mathcal{O}(n^2)$. An upper bound $\hat{y}$ is given by performing local descent using a discrete analogue of a gradient [168]. Having an initial runtime of $\mathcal{O}(n^3)$ it also can be dynamically updated making it computable in $\mathcal{O}(n^2)$. This leads to an initial computational effort of $O(n^3)$ and $O(n^2)$ for every subsequent branched state.

Having $y^-(\boldsymbol{Q})$ and $y^+(\boldsymbol{Q})$ at hand, h can be computed in $\mathcal{O}(n)$. This leads to a total computational cost of $\mathcal{O}(n^2)$ for a single state. Thus, every upper bound $\hat{r}(\boldsymbol{Q}, T)$ (policy rollout) can be computed in $\mathcal{O}(Tn^2)$.

For the computational complexity of the lower bound $\check{r}(\boldsymbol{Q}, T)$, we first consider the lower bound $\check{b}(\boldsymbol{Q}, T)$. Initially, the parameters of $\boldsymbol{Q}$ and the elements in $\bar{\mathcal{D}}$ can be sorted in $\mathcal{O}(n^2 \log(n))$. Updating single parameters, this sorting can be dynamically updated in $\mathcal{O}(n^2)$. For $\hat{b}(\boldsymbol{Q}, T)$, this is repeated T times, leading to a computational effort of $\mathcal{O}(Tn^2)$. The bound $\check{b}(\boldsymbol{Q}, T)$ can be computed in $\mathcal{O}(T)$. Thus, the computational complexity of the lower bound $\check{r}(\boldsymbol{Q}, T)$ is $\mathcal{O}(Tn^2)$. Combined with the runtime for computing an upper bound $\hat{r}(\boldsymbol{Q}, T)$ results in a total runtime $\mathcal{O}(Tn^2)$ of the bound-step.

4.4 Experimental Evaluation

In this section we conduct experiments to demonstrate the effectiveness of our proposed method.

4.4.1 Greedy Policies for Random Instances

We first conduct experiments comparing two base policies, called π^R and $\dot{\pi}^R$. The policy π^R randomly chooses an index pair from $[n] \times [n]$. $\dot{\pi}^R$ chooses the next indices greedily according to $\arg\min_{a \in \mathcal{A}} \mathsf{DR}\left(f(\boldsymbol{Q}, a)\right)$ and if it gets stuck in a local optimum, it switches to π^R. Different heuristics are compared for updating the parameter in the transition function $f(\boldsymbol{Q}, (k, l)) = \boldsymbol{Q} + h(\boldsymbol{Q}, (k, l))\boldsymbol{e}_k \boldsymbol{e}_l^\top$ which were presented in Section 4.3, namely

- G: Greedily choose the parameter update that leads to the greatest DR decrease;
- G_0: Like G, but prefer to set parameters to 0, if possible;
- M: Restrict bounds such that the parameter ordering remains intact.

The methods described are implemented as part of our Python package `qubolite`[3].

Using 1000 random QUBO matrices with the entries being sampled uniformly in the interval $[-0.5, 0.5]$, we apply the different heuristics for 1000 iterations. For preserving the optimum, the upper bounds $\hat{y}_{ab}$ are computed using a local search, and the lower bounds $\check{y}_{ab}$ are computed by using the roof-dual algorithm [171]. The 95%-confidence intervals are indicated.

Figure 4.8 shows the DR ratio between the matrices arising by following the different heuristics and the original QUBO matrix, that is $\mathsf{DR}(\boldsymbol{Q} + \boldsymbol{A})/\mathsf{DR}(\boldsymbol{Q})$.

We observe this ratio to be monotonically decreasing, indicating that our proposed heuristics are fulfilling their duty of reducing the DR. This decreasing percentage shrinks with larger n, i. e., it is

[3] https://github.com/smuecke/qubolite (last accessed September 19, 2025)

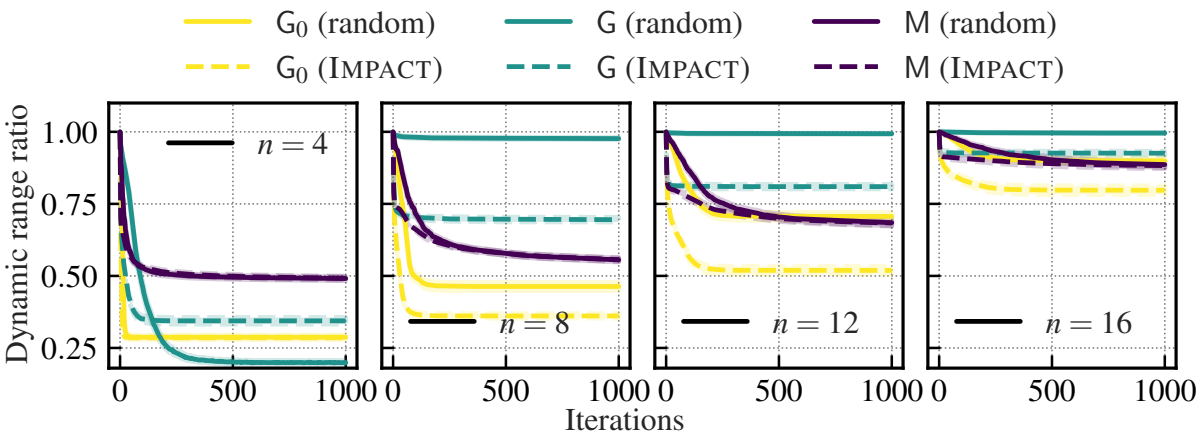


Figure 4.8: Dynamic range ratio for random QUBO instances with sizes $n \in \{4, 8, 12, 16\}$.

harder to reduce the DR for larger n. Generally, choosing the next QUBO weight according to the myopic IMPACT on the DR is better than a random decision (except for $n = 4$), but also comes with a higher computational cost. With increasing n, it turns out that the G heuristic performs worse than the M heuristic. However, the slight adaption G_0 of setting QUBO weights to 0 if possible outperforms M.

After relatively few iterations the DR ratio seems to converge to a minimal value for all methods we investigate. Convergence is faster for small n. Also with increasing n, the minimum DR ratio approaches one, i. e., a smaller overall improvement can be achieved. We suspect that the iterative procedure reaches a local optimum where none of the weights can be reduced without changing the minimizing vector, according to the bounds employed. Naturally, this happens faster with fewer weights, i. e., with smaller n, on which the number of weights depends according to $\mathcal{O}(n^2)$.

Convergence could be improved by using sharper bounds, which comes at higher computational cost, though, e. g., computing the roof dual bound has time complexity $\mathcal{O}(n^3)$. We expect that clever implementations of bounds can exploit the fact that only one weight at a time is changed, allowing for re-use of data structures (like the flow network constructed for the roof dual bound), reducing the computation time drastically.

For further evaluation, we need to define the notion of the *induced ranking* of a QUBO instance, and a way to compare these rankings.

Definition 4.9. Let $\boldsymbol{Q} \in \mathcal{Q}_n$, and let $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{2^n}$ be the binary vectors of $\mathbb{B}^n$ in lexicographical order. The *induced ranking* of $\boldsymbol{Q}$ is a permutation $\pi_{\boldsymbol{Q}} \in [2^n] \to [2^n]$ such that

$$E_{\boldsymbol{Q}}(\boldsymbol{z}_{\pi_{\boldsymbol{Q}}(i)}) \leq E_{\boldsymbol{Q}}(\boldsymbol{z}_{\pi_{\boldsymbol{Q}}(i+1)}) \; \forall i \in [2^n - 1] \; .$$

Definition 4.10. Let $\pi, \pi' : [K] \to [K]$ be two permutations for some $K > 1$. The normalized *Kendall* τ *distance* between π and π' is given by

$$K_d(\pi, \pi') \coloneqq \frac{1}{2} + \frac{1}{K(K-1)} \sum_{\substack{i,j \in [K] \\ i<j}} \operatorname{sign}\left[(\pi(i) - \pi(j)) \cdot (\pi'(i) - \pi'(j))\right] \; .$$

Intuitively, this distance measures the proportion of disagreement between rankings over all pairs

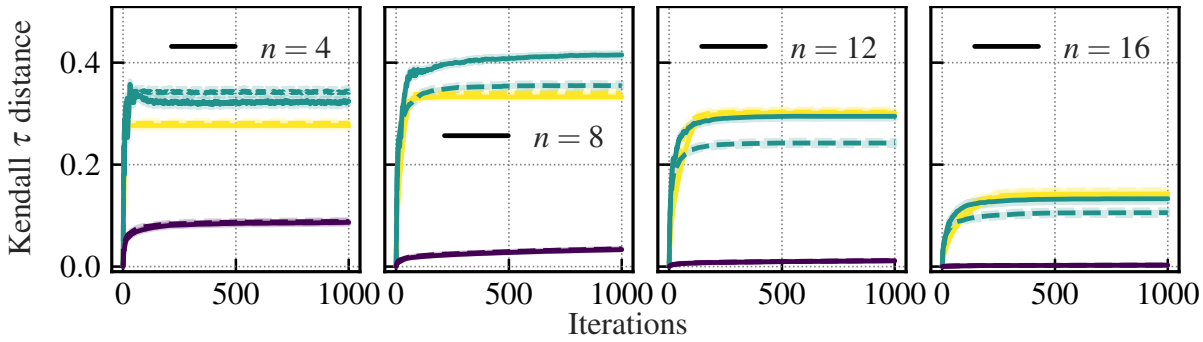


Figure 4.9: State ordering for random QUBO instances with sizes $n \in \{4, 8, 12, 16\}$.

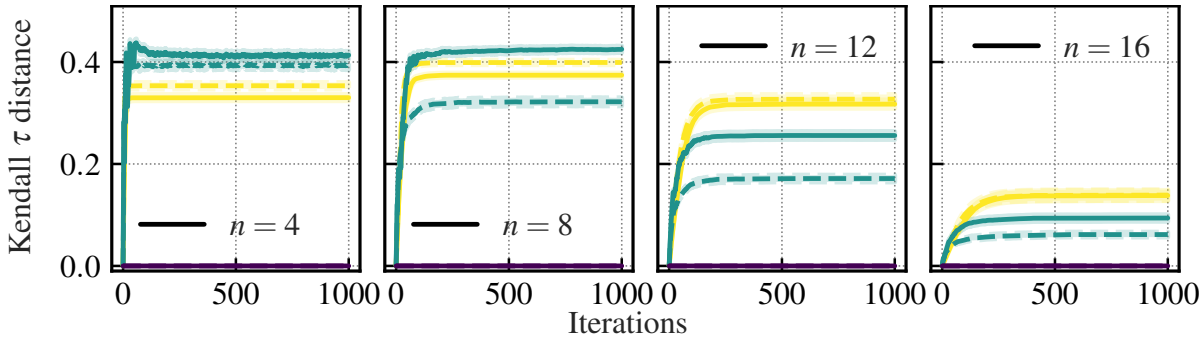


Figure 4.10: Distance between parameter orderings for random QUBO instances with sizes $n \in \{4, 8, 12, 16\}$. Note that the line for method M (random) is constantly 0.

of indices, i. e., the percentage of (i, j) such that $\pi(i) < \pi(j)$, but $\pi'(i) > \pi'(j)$, and vice versa. If $K_d(\pi, \pi') = 0$, then π and π' are identical, and if $K_d(\pi, \pi') = 1$, then π' is the reverse of π. Therefore, Kendall τ distance gives us a measure of how much the DR reduction "scrambles" the value landscape of $f_{\boldsymbol{Q}}$ by computing $K_d(\pi_{\boldsymbol{Q}}, \pi_{\boldsymbol{Q}+\boldsymbol{A}})$, which is relevant, e. g., for the performance of local search heuristics.

Figure 4.9 shows the Kendall τ distance between the induced rankings of the QUBO instances on $\mathbb{B}^n$ before and after their DR reduction. We observe that G and G_0 "scramble" the ordering of binary vectors more strongly than M. This is exactly what we expected, as M's objective is to preserve the ordering of parameters, which in turn leads to more conservative parameter modifications. This reduces the overall "noise" that is added to the energy landscape.

Instead of considering the QUBO energy values themselves, we plot the change of the QUBO weight ordering in Figure 4.10. Again, the Kendall τ distance is used and we observe similar effects to Figure 4.9. We can see that M maintains the weight ordering.

Finally, Figure 4.11 displays the unique weight ratio, that is the number of unique QUBO weights of the current iteration divided by the number of unique QUBO weights of the original QUBO. M never changes the uniqueness of the weights, while the ratio is monotonically reduced for G^0, since many weights are set to 0. With using G, the number of unique weights is first reduced but then starts to increase when the to be changed weights are chosen randomly.

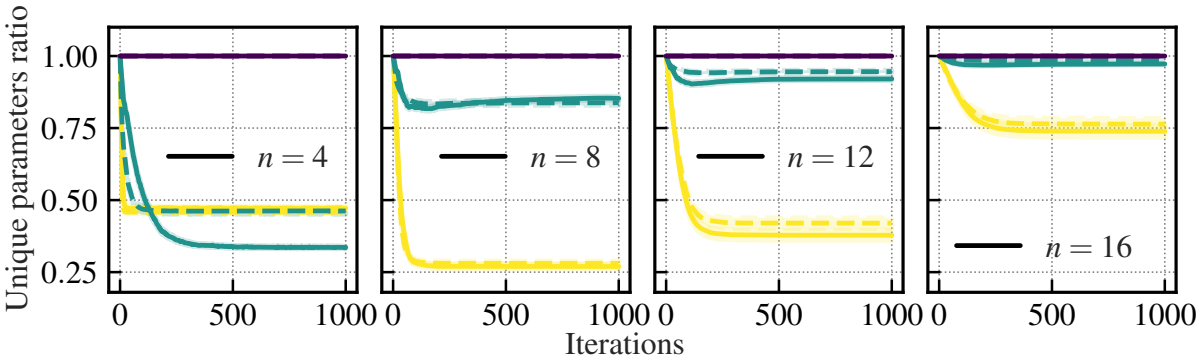


Figure 4.11: Percentage of unique parameter values for random QUBO instances with sizes $n \in \{4, 8, 12, 16\}$. Note that the line for method M (random) is constantly 1.

4.4.2 Data Dependent QUBO Embeddings

We want to investigate to what extend our DR reduction method can help to improve the performance of actual Ising machines, such as NISQ devices.

As three exemplary problems, we perform BICLUS, VECQUANT and SUBSUM. BICLUS stands for "binary clustering" and is an unsupervised ML task, where data points are assigned to one of two classes ("clusters"). SUBSUM consists of finding a subset from a list of values that sum up to a given target value. Both have well-established QUBO embeddings [59, 173]. We chose these problems as they are (i) real-world problems of both scientific and economic interest, (ii) easy to generate for arbitrary n, and (iii) their QUBO instances' DR is a direct result of the input data. The last point is especially important, as the DR has a tangible connection to the underlying data, which we can generate to obtain realistic QUBO instances with high DR.

BICLUS To generate data for BICLUS, we sample i.i.d. $n = 20$ 2-dimensional points from an isotropic standard normal distribution. Then we create two clusters by applying $(x_1, x_2) \mapsto (x_1 - 4, x_2)$ to the first ten points, and $(x_1, x_2) \mapsto (x_1 + 4, x_2)$ to the last ten. As a last step, we choose the points 1 and 19 and multiply their coordinates by 20, which leads to a data set containing two outliers.

From this data we derive a QUBO instance $\boldsymbol{Q}$ given in Proposition 3.3

$$\min_{\boldsymbol{z} \in \mathbb{B}^n} -\boldsymbol{z}^\top \boldsymbol{K} \boldsymbol{z} + \mathbf{1}^\top \boldsymbol{K} \boldsymbol{z} \,,$$

where $\boldsymbol{K}$ is a kernel matrix. We use a linear kernel, which leads to a vanilla 2-means clustering based on Euclidean distance. To ensure this problem has only one optimal solution, we assign class 0 to point 20 and only optimize over the remaining 19 points (otherwise there would be two symmetrical solutions).

SUBSUM For the SUBSUM problem, we are given a set $\mathcal{A} = \{a_1, \ldots, a_n\} \subset \mathbb{Z}$ and $T \in \mathbb{Z}$. The goal is to find $\mathcal{I} \subset [n]$, s.t., $\sum_{i \in \mathcal{I}} a_i = T$. We use the same problem instance as is given in [45], Section 4.2. We set $n = 16$ and generate the elements of $\mathcal{A}$ as $||\lfloor 10 \cdot Z \rceil||$, where Z follows a standard Cauchy distribution. This approach leads to occasional outliers with large magnitudes, which in turn produced QUBO instances with a high degree of difficulty due to large DR. Next, we determined the number of summands k by sampling from a triangular distribution $\lfloor U \rceil$, where U is defined with parameters $a = \frac{n}{5}$,

$b = \frac{n}{2}$, and $c = \frac{4n}{5}$, ensuring that, on average, half of the elements of $\mathcal{A}$ contribute to the sum. Finally, we selected k indices from $[n]$ without replacement to form the subset $\mathcal{I}$ and set $T = \sum_{i \in I} a_i$, thereby creating problems where the global optimum is predetermined.

We obtain a QUBO formulation, where we use n binary variables which indicate if $i \in S$ for each i. With $\boldsymbol{a} = (a_1, \ldots, a_n)$, a QUBO formulation is given by

$$\min_{z \in \{0,1\}^n} \left(\boldsymbol{a}^\top \boldsymbol{z} - T \right)^2 \Leftrightarrow \min_{z \in \{0,1\}^n} \boldsymbol{z}^\top \boldsymbol{a}^\top \boldsymbol{a} \boldsymbol{z} - 2T \boldsymbol{a}^\top \boldsymbol{z} \ .$$

VECQUANT VQ deals with the problem of finding prototypes of a given set of vectors, which give a best representation according to some measure. We use the approach from Proposition 3.4, where the goal is to find k medoids according to the well known k-medoids objective function. A QUBO formulation is given by

$$\min_{\boldsymbol{z} \in \{0,1\}^n} \boldsymbol{z}^\top \left(\gamma \boldsymbol{1}\boldsymbol{1}^\top - \alpha \boldsymbol{D} \right) \boldsymbol{z} + \left(\beta \boldsymbol{D}\boldsymbol{1} - 2\gamma k \boldsymbol{1} \right)^\top \boldsymbol{z} \ ,$$

where $\boldsymbol{D}$ is a pairwise distance matrix for, α is a weight for identifying far apart data points, β is for identifying central data points and γ ensures that we choose exactly k vectors. We follow [58] and set $\alpha = 1/k$, $\beta = 1/n$ and use Welsh's distance

$$d(\boldsymbol{x}, \boldsymbol{y}) \coloneqq 1 - \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2} \right) \ ,$$

for computing $\boldsymbol{D}$. The penalty parameter γ, which enforces that exactly k prototypes are chosen, is set to 2. For hardware evaluation, we use the same dataset as BICLUS and set $k = 4$.

We also compare the results of our proposed methods with two other methods from the literature.

Adding Auxiliary Variables In [157], a method is proposed which reduces the bit-width of single parameters of the underlying Ising model. It adds auxiliary variables to the problem in a way such that an optimal configuration still corresponds to an optimum of the original problem. It is assumed that all problem parameters are integer, i.e., $\boldsymbol{J} \in \mathbb{Z}^{n \times n}, \boldsymbol{h} \in \mathbb{Z}^n$. If we want to change a parameter J_{ij}, $i \neq j$ to reduce its bit-width, we can do this by introducing a new variable x to the problem to obtain $\boldsymbol{J}' \in \mathbb{Z}^{n+1 \times n+1}$ such that

$$J'_{ij} = J_{ij} - r, \ J'_{ix} = |r|, \ J'_{xj} = -r \ .$$

If we want to change h_i, optimality is ensured by

$$h'_i = h_i - r, \ h'_{ix} = -|r|, \ h'_x = r \ .$$

That is, if $\bar{\boldsymbol{s}} \coloneqq \arg\min_{\boldsymbol{s} \in \{-1,1\}^{n+1}} \boldsymbol{s}^\top J' \boldsymbol{s} + \boldsymbol{s}^\top \boldsymbol{h}'$, then

$$\bar{\boldsymbol{s}}_{[n]}^\top \boldsymbol{J} \bar{\boldsymbol{s}}_{[n]} + \bar{\boldsymbol{s}}_{[n]}^\top \boldsymbol{h} = \min_{\boldsymbol{s} \in \{-1,+1\}^n} \boldsymbol{s}^\top J \boldsymbol{s} + \boldsymbol{s}^\top \boldsymbol{h} \ ,$$

where the subscript $[n]$ denotes the vector consisting of the first n entries. However, to reduce the bit-width by m bits, $\mathcal{O}(n2^m)$ auxiliary variables are introduced. Due to limited capability of hardware solvers in terms of representable problem size, this can pose a problem on finding a solution.

In our experiments we apply this method to SUBSUM, since our datasets are integer. We reduce the bit-width of our parameters by 2, since no further benefit on the performance of hardware solvers was observed for introducing more variables.

Tuning Penalty Parameters As a second baseline, we use the method from [146]. It is assumed that the optimization problem is given in a quadratic binary constrained form

$$\min_{\boldsymbol{z}\in\{0,1\}^n} \boldsymbol{z}^\top \boldsymbol{Q}\boldsymbol{z} \quad \text{s.t. } \boldsymbol{A}\boldsymbol{z} = \boldsymbol{b} \ ,$$

which can be brought in an equivalent QUBO form by introducing a penalty parameter $\lambda > 0$

$$\min_{\boldsymbol{z}\in\{0,1\}^n} \boldsymbol{z}^\top \boldsymbol{Q}\boldsymbol{z} + \lambda(\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b})^\top(\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}) \ .$$

This parameter has to be chosen large enough to ensure equivalence. We use the method

$$\lambda = \hat{\boldsymbol{z}}^\top \boldsymbol{Q}\hat{\boldsymbol{z}} - \check{E}_{\boldsymbol{Q}} \ ,$$

where $\hat{\boldsymbol{z}}$ is a feasible solution, i.e., $\boldsymbol{A}\hat{\boldsymbol{z}} = \boldsymbol{b}$ and $\check{E}_{\boldsymbol{Q}}$ is a lower bound on the optimum, i.e., $\check{E}_{\boldsymbol{Q}} \leq \min \boldsymbol{z}^\top \boldsymbol{Q}\boldsymbol{z}$. For increasing λ the DR is also increased, thus choosing λ as small as possible is favourable. Hence, we use the exact solutions for computing λ in our experiments. However, this is intractable in realistic scenarios.

We denote the method of tuning penalty parameters for incorporating constraints in [146] as PEN and the method of introducing auxiliary variables to reduce the BW [157] as AUX. However, they are not generally applicable to arbitrary QUBO problems, i.e., PEN can only be applied to VECQUANT since it incorporates a constraint of finding exactly k prototypes and AUX can only be applied to SUBSUM, since we here use QUBO instances with integer values. However, our method can be applied to arbitrary QUBO problems.

4.4.3 Performance of Branch-and-Bound

In what follows, we consider numerical experiments and study the impact of our method on a D-Wave Advantage System 5.4 QA device and FPGA-based DA hardware [71] in Section 4.4.4. Three exemplary problems are considered: BICLUS represents 2-means clustering, SUBSUM consists of finding a subset from a list of values that sum up to a given target value and VECQUANT aims for finding prototype vectors. All three problems have known QUBO embeddings [58, 59, 173], discussed in the previous section.

We compare different policies: the base policy $\dot{\pi}$ (heuristic baseline discussed in Section 4.3.1), our B&B policy $\bar{\pi}_{\tilde{T}}$ with different rollout horizons $\tilde{T}$ and our rollout selection policy $\tilde{\pi}$. The relative DR reduction for a horizon up to $T = 10$ can be found in Figure 4.12. We compare ALL (left) and IMPACT (right) for choosing the indices in the branch step. It is apparent that every single policy reduces the DR with an increasing horizon T. The base policy $\dot{\pi}$ is largely outperformed by our $\bar{\pi}_{\tilde{T}}$ and $\tilde{\pi}$. $\bar{\pi}_{\tilde{T}}$ is increasing its performance with an increasing rollout horizon $\tilde{T}$. We can see that the exact method ALL has the same performance as using the simplified version IMPACT, while being more computational demanding. It scales quadratically with the problem size n, where IMPACT is basically independent of n. The policy $\tilde{\pi}$ already almost achieves optimal performance (c.f. to $\bar{\pi}_{10}$).

We also evaluate our results with the coefficient ratio (CR) (Equation (4.3)) for $n = 8$. It is evident from Figure 4.13, that $\log_2(\mathsf{CR}(\boldsymbol{Q})) \leq \mathsf{DR}(\boldsymbol{Q})$, aligning with our theoretical insights.

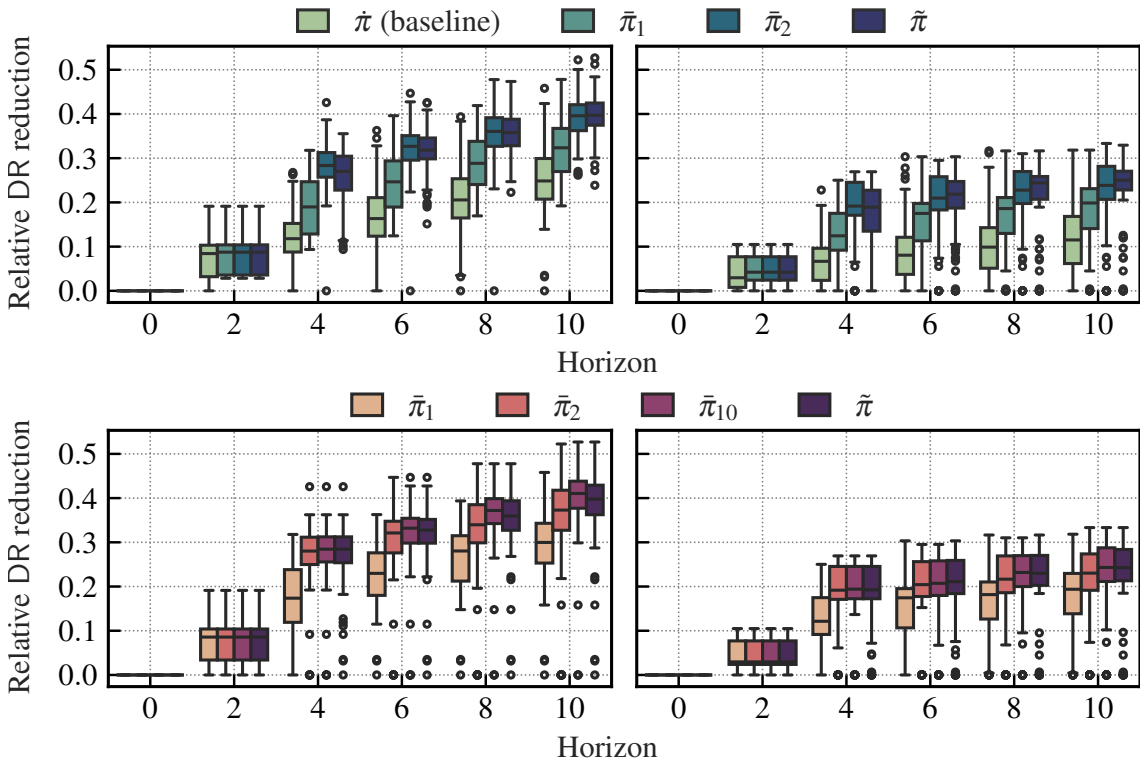


Figure 4.12: Relative DR reduction for 100 BICLUS instances with $n = 8$ (first and third column plot) and $n = 16$ (second and fourth column plot). Different policies are compared for choosing the indices with ALL (left) and IMPACT (right).

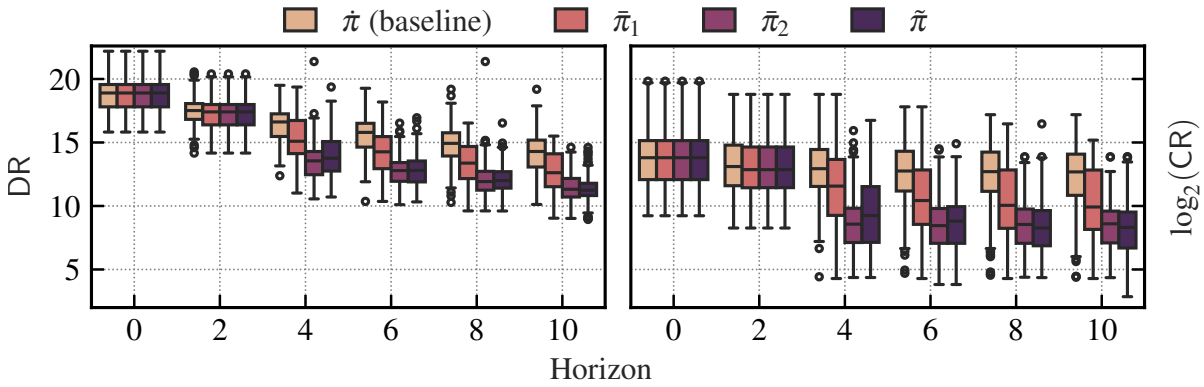


Figure 4.13: Comparison between absolute DR and $\log_2(\mathrm{CR})$ reduction for $n = 8$ and IMPACT indices.

Evaluating the quality of our bounds (see Section 4.3.4), we indicate the fraction of the pruned state space in Figure 4.14. We here consider the exact solution, that is $\bar{\pi}_T$. We vary the depth until the upper bounds are updated. Increasing this depth, as well as increasing the horizon leads to pruning a larger fraction of the whole search space. The number of pruned states does not heavily depend on an updated current best, indicating the strength of our lower bound (Section 4.3.4).

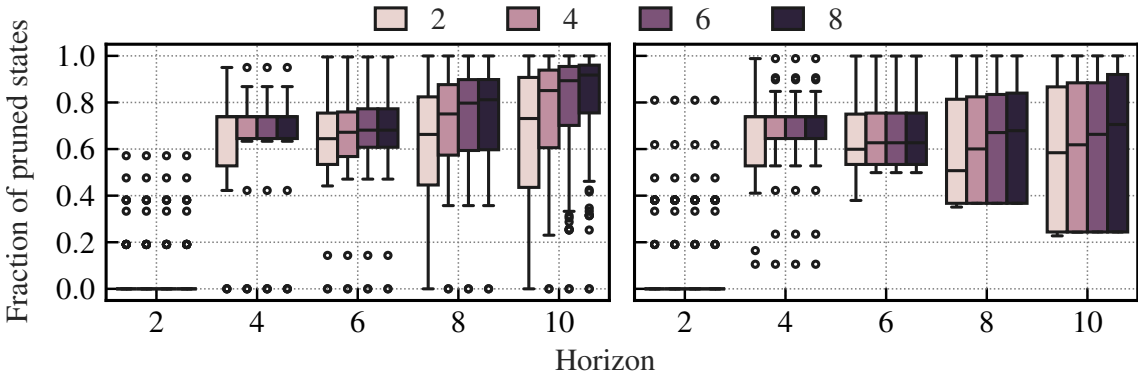


Figure 4.14: Fraction of pruned states. Different depths (2, 4, 6 and 8) for updating the current best DR are compared for $n = 8$ (left) and $n = 16$ (right).

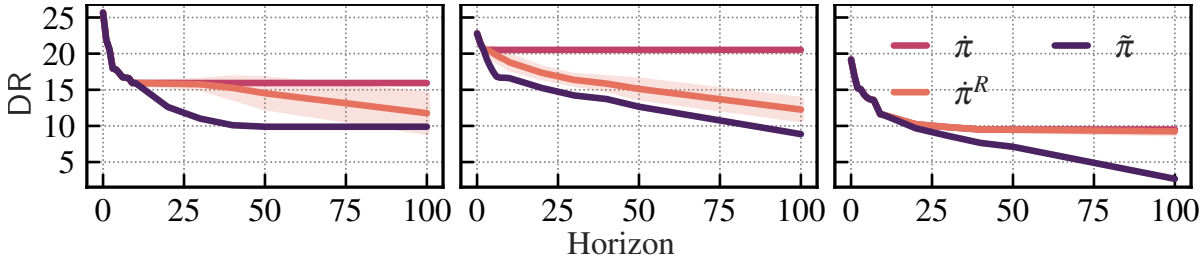


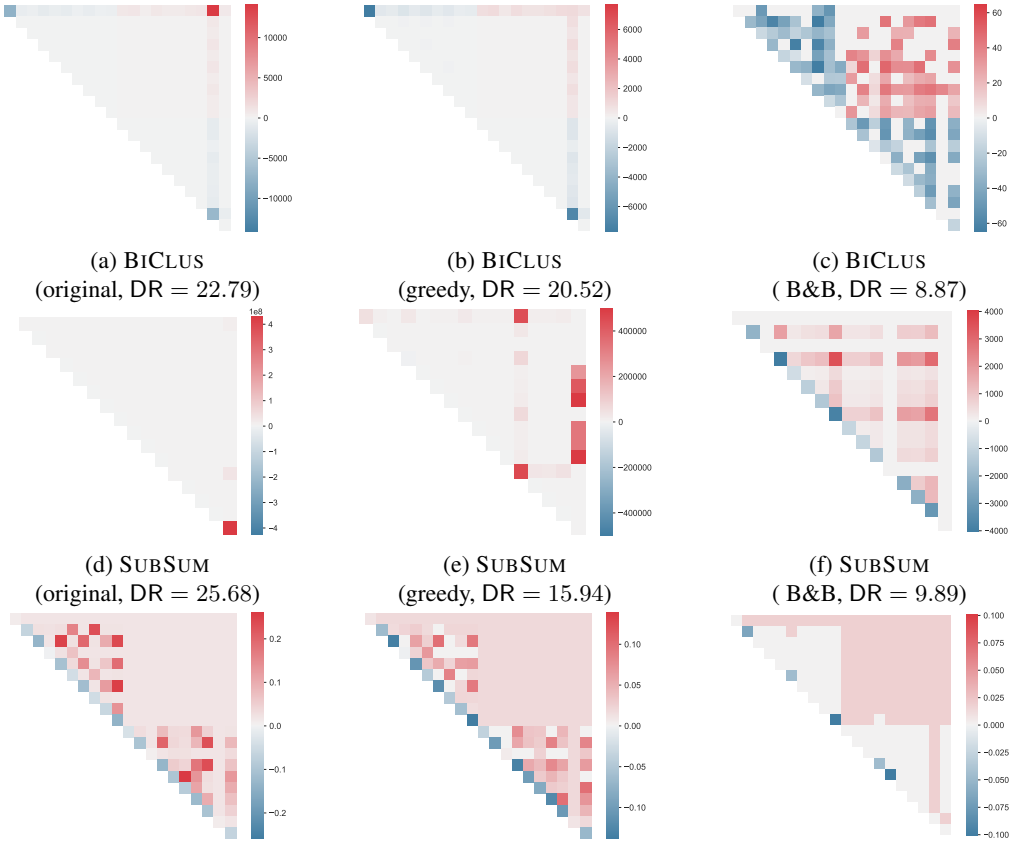
Figure 4.15: Performance of our developed policy $\tilde{\pi}$ compared to the base policy $\dot{\pi}$ and the randomized base policy $\dot{\pi}^R$. The DR reduction is compared for a SUBSUM (left), a BICLUS (middle) and a VECQUANT (right) instance.

Table 4.1: Comparison of QUBO hardware solvers using different methods (details in Figure 4.17). We depict the DR along with the number of optimal samples (from 1000) obtained by QA and DA with 16, 8 and 4 bit precision. Optima are bold and dashes indicate that the method is not applicable for the respective problem.

	SUBSUM					BICLUS					VECQUANT				
	DR	QA	DA16	DA8	DA4	**DR**	QA	DA16	DA8	DA4	**DR**	QA	DA16	DA8	DA4
Orig	25.68	0	5	1	0	22.79	3	**1000**	**1000**	1	19.19	18	1000	1000	0
GRE	15.94	3	687	**3**	0	20.52	605	462	0	0	9.51	1	1000	0	0
AUX	25.68	0	7	2	**2**	—	—	—	—	—	—	—	—	—	—
PEN	—	—	—	—	—	—	—	—	—	—	24.63	0	1000	1	0
Ours	**9.89**	**26**	**1000**	0	0	**8.87**	**865**	585	579	**528**	**2.68**	**356**	**1000**	**1000**	**487**

Further, we compare the DR reduction performance of the base policy $\dot{\pi}$ and a randomized base policy $\dot{\pi}^R$ with our rollout selection policy $\tilde{\pi}$ in Figure 4.15 for three different instances of SUBSUM, BICLUS and VECQUANT of dimensions $n = 16, 20, 20$. We can see that $\dot{\pi}$ ends up in local optima pretty fast while our $\tilde{\pi}$ is more robust and is steadily improving with an increasing horizon. It also outperforms $\dot{\pi}^R$, which needs a lot of iterations for an increasing QUBO dimension. Specifics on the obtained DR can be found in Table 4.1—our method always achieves the smallest DR.

The parameter values of the original are shown in the left column of Figure 4.16. As is apparent



(a) BICLUS (original, DR = 22.79)
(b) BICLUS (greedy, DR = 20.52)
(c) BICLUS (B&B, DR = 8.87)
(d) SUBSUM (original, DR = 25.68)
(e) SUBSUM (greedy, DR = 15.94)
(f) SUBSUM (B&B, DR = 9.89)
(g) VECQUANT (original, DR = 19.19)
(h) VECQUANT (greedy, DR = 9.51)
(i) VECQUANT (B&B, DR = 2.68)

Figure 4.16: QUBO parameter matrices for BICLUS, SUBSUM and VECQUANT problems, original (left), using the greedy policy (middle) and applying our B&B algorithm (right) for 100 iterations. Notice that the difference in DR is illustrated by the color scale, which renders most parameter values in the original QUBO matrices indistinguishable, which is greatly improved, especially on the right.

from the color scale, the DR is very high, which makes most values near 0 completely indistinguishable. We apply our base policy $\dot{\pi}$ and the B&B rollout policy $\tilde{\pi}$ with a fixed budget of 100 iterations. The resulting QUBO instances are shown in the middle and right column, respectively. Clearly, much more detail is visible, as the color scale is much narrower, already hinting at the lower DR. While the base policy can get stuck in local optima pretty fast, our proposed B&B largely reduces the DR

4.4.4 Performance on Hardware Solvers

Further, we assess the impact of the reduced QUBO instances on two hardware QUBO solvers (Ising machines)—QA and DA. The D-Wave annealers have a fixed connectivity structure, i.e., only a subset of qubit pairs can be assigned a weight. Therefore, dense QUBO problems (like ours) must be embedded into this connectivity graph structure through redundant encoding and additional constraints. To improve comparability, we have this embedding computed once for the original QUBO and re-use it for the compressed one.

Further, NISQ devices are prone to *Integrated Control Errors* which constrain the DR of the hardware parameters. If the DR of the given QUBO is too large, it can happen that a completely different problem is solved due to implementation noise of the parameters. A similar problem appears for hardware solvers with a fixed bit precision: the parameters have to be rounded/quantized to that precision, which can lead to different optima (see Figure 4.1). Furthermore, DA designs can be improved by considering a reduced bit precision. In our experiments, the hardware plattform for the digital annealer is an AMD Virtex UltraScale+ FPGA VCU118 evaluation board.

By implementing the digital annealer chip design with AMD Vivado for 16, 8 and 4 bit precision, we find that when using 4 instead of 16 bit precision, the number of on-chip signals is reduced by 28.29%. This has multiple benefits: First, the improvement allows us to run the orginal design with a reduced power consumption when operating the hardware solver. E.g., the power requirement for on-chip memory[4] shrinks from $1.4\,\mathrm{W}$ to $0.3\,\mathrm{W}$. Second, due to less occupied chip space, the reduction also allows for an increase of the maximum number of QUBO problem variables. However we did not evaluate this option as it requires significant changes of the chip design which are out of scope of our study.

Now, after adressing the resource consumption, we answer the question whether reducing the DR leads to an optimized performance for QA and DA. To quantify the improvement, we consider the energy values of samples obtained from the compressed QUBO instances w.r.t. the original (uncompressed) QUBO. Due to their probabilistic nature, we generate 1000 *samples* and use default parameters. For DA, we use three different bit precisions of the internal arithmetics, i.e., 16, 8 and 4 bit. For making the performances comparable we evaluate the original QUBO energy $E_{\boldsymbol{Q}}(\boldsymbol{z})$ for every sample $\boldsymbol{z}$ and examine the relative distance to the optimum energy $v_{\boldsymbol{Q}}(\boldsymbol{z}) \coloneqq (E_{\boldsymbol{Q}}(\boldsymbol{z}) - E_{\boldsymbol{Q}}(\boldsymbol{z}^*))/E_{\boldsymbol{Q}}(\boldsymbol{z}^*)$. This is depicted in Figure 4.17, where we indicate the energy distribution for the initial QUBO matrix $\boldsymbol{Q}$, the base policy $f_{100}(\boldsymbol{Q}, \dot{\pi})$, AUX for SUBSUM, PEN for VECQUANT and our rollout selection policy $f_{100}(\boldsymbol{Q}, \tilde{\pi})$. Even though our DR reduction method can change the overall energy landscape, we are interested in the low energy values since we aim to minimize the energy. In Table 4.1, we depict the total number of optimal samples from the 1000 drawn samples, using different methods for QA and DA. We can see that our method almost always outperforms the baselines in terms of ability of the hardware solver to find optimal solutions. We conclude that our method helps a quantum annealer and FPGA-based technology to find the optimum more reliably.

4.5 Conclusion

In this chapter, we expanded our investigation beyond the spectral gap to consider the broader factors influencing QO performance, particularly in the presence of noise and hardware limitations. We demonstrated how finite numerical precision and inherent hardware imperfections can lead to deviations from the true optima, posing significant challenges for both quantum and classical Ising machines. The proneness of QUBO matrices for such errors is quantized by the DR, indicating the number of bits needed to faithfully encode their parameters. Finding an optimum preserving QUBO matrix with minimal DR, however, is as hard as solving the QUBO itself. To address these issues, we introduced a principled MDP framework for analyzing error susceptibility in QUBO instances and developed an iterative methodology for reducing the DR of problem matrices. Our proposed precision-reduction

[4] Block RAM and Ultra RAM combined. See https://docs.amd.com/v/u/en-US/ug573-ultrascale-memory-resources (last accessed September 19, 2025). Numbers reported here are estimated by the FPGA design software.

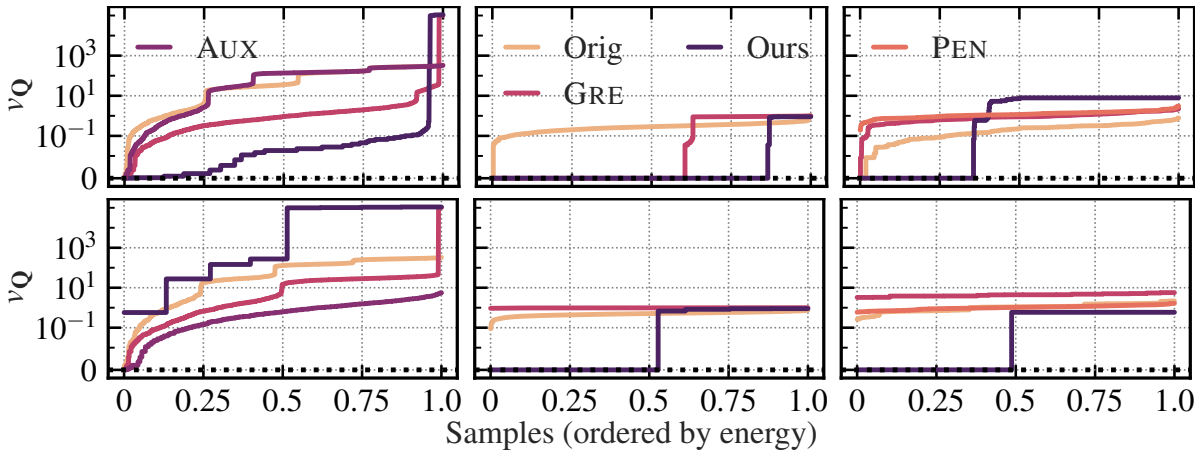


Figure 4.17: Performance of the D-Wave Advantage 5.4 (top row) and an FPGA-based digital annealer with 4-bit precision (bottom row): we compare the original QUBO $\boldsymbol{Q}$ (Orig), the QUBO using the greedy base policy (GRE) for 100 steps $f_{100}(\boldsymbol{Q}, \dot{\pi})$, the method in [146] (PEN) for computing the penalty, the method of adding auxiliary variables (AUX) [157] and our rollout selection policy for 100 steps $f_{100}(\boldsymbol{Q}, \tilde{\pi})$. The relative energies $v_{\boldsymbol{Q}}$ for 1000 samples are shown for a SUBSUM (left), a BICLUS (middle) and a VECQUANT (right) instance.

techniques, ranging from heuristic approaches to an exact B&B algorithm, offer a systematic way to mitigate the impact of noise while preserving the integrity of optimization results. For this, we established upper and lower bounds, which allowed us to compute intervals in which parameter values can be modified without affecting the minimizing vectors, leaving optimal solutions intact. Further, we proposed computationally efficient and theoretically sound bounds for pruning, leading to drastic reduction of the search space size. We combined our B&B approach with the well-known policy rollout for improving computational efficiency and the performance of already existing heuristics.

Through extensive experimentation on ML-related CO problems, we established a clear link between data properties and the required numerical precision, reinforcing the necessity of precision-aware preprocessing. We use our method to reduce the DR of NP-hard real-word problems, such as clustering, subset sum and VQ. Our proposed algorithm largely outperforms recently developed algorithms, while also being applicable to arbitrary QUBO problems. The effectiveness for hardware solvers is shown for a real quantum annealer and an FPGA-based digital annealer. We conclude that our method enhances the reliability of QA and DA in finding the optimum, and reduces the power consumption of DA. The findings suggest that optimizing QUBO representations not only enhances robustness in NISQ era quantum devices but also provides measurable improvements in classical hardware solvers, including FPGA-based annealers. In summary, we provide an efficient pre-processing technique that can be applied out-of-the-box to arbitrary QUBO instances to improve their feasibility.

In future work, it would be interesting to investigate further techniques for approximate dynamic programming, such as reinforcement learning. Furthermore, evaluation of the effect of our algorithm for using different hardware, such as GPUs, or coherent Ising machines is intriguing. Another factor is the qubit topology of the NISQ hardware. For our experiments, we used the same qubit mapping for both the original and compressed QUBO instances, in order to eliminate this variable when comparing the result quality. However, it could be well worth adapting the compression methods presented here to actively improve qubit allocation, e. g., by preferably setting weights to 0 that would otherwise lead to longer chain lengths. The performance of DR reduction on native-connectivity QUBO instances could

moreover be an interesting benchmark.

In this first part of the thesis—Chapter 3 and this chapter—we found that the performance of NISQ devices for QO tasks is highly influenced by problem data properties, particularly in ML applications where data separability and compactness affect solution quality. Beyond noise, these devices face constraints like a limited number of qubits and sparse connectivity, making it difficult to implement large-scale QUBO problems efficiently. The high dimensionality of ML-related CO problems exacerbates these challenges, leading to increased gate depth and error accumulation. In the upcoming second part (Chapters 5 to 7), we address these limitations with problem size reduction techniques, such as problem decomposition, variable generation and using different QUBO encodings.

Part II

Effects of Data Scale on Quantum Optimization

CHAPTER 5

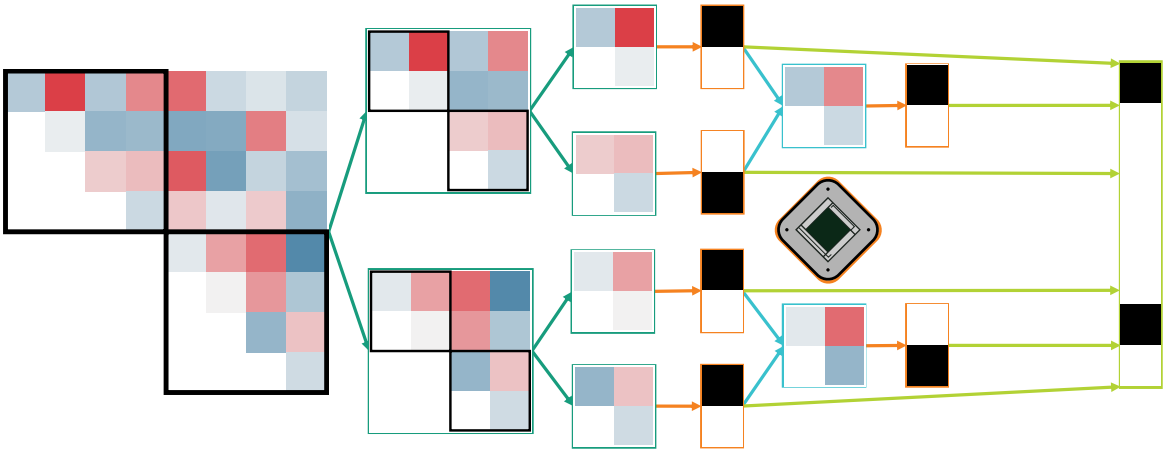
Recursive QUBO Decomposition

As we have seen in Part I of this thesis, the performance of NISQ devices for QO tasks is largely dependent on the underlying data complexity. Especially in typical ML tasks, the data separability or compactness can play a large role for the ability to find an optimal solution. This is, inter alia, due to the proneness to errors in quantum computations. However, noise is not the only issue of current quantum hardware, since NISQ devices also suffer from a limited number of qubits. Many CO problems for ML have an enormous problem dimensionality, either through the size of the dataset or the number of features. These constraints make it challenging to implement large-scale QUBO problems efficiently, thus necessitating techniques to reduce problem size while preserving solution quality. NISQ devices typically contain only tens to a few hundred qubits, which limits the direct encoding of large QUBO instances. Moreover, the connectivity between qubits in current hardware is sparse, requiring additional qubits and operations to implement problem constraints effectively. This leads to increased gate depth and error accumulation, which further degrades computational performance. Given these limitations, problem size reduction methods—such as decomposition techniques, variable reduction heuristics, and hybrid quantum-classical approaches—are essential for optimizing QUBO formulations.

In this second part, we investigate different techniques for reducing the intractable QUBO problem size, which stem from the underlying data. This not only enables practical implementations on NISQ devices but also improves solution fidelity by minimizing the impact of quantum noise. Specifically, problem decomposition allows for breaking down large QUBO instances into smaller subproblems that can be solved independently or iteratively, as described in this chapter. Chapter 6 is concerned with iteratively adding constraints and variables to the problem, leading to a well-controllable problem size. A different approach is taken in Chapter 7, where we consider techniques to refine an already available feasible solution by using efficient QUBO encodings, instead of considering subproblems.

Reasons behind a large problem size can be manifold—a great dataset size, a huge amount of features or the need for incorporating many constraints. In Section 5.2, we examine how to integrate linear inequality constraints into the objective of a QBP problem to obtain a QUBO formulation. Since this can result in an intractable problem size for NISQ devices, we consider techniques of iteratively solving subproblems, converging towards the optimal solution in Section 5.3. However, it can be hard to choose the subproblem structure, especially in the face of with constraints concerning many variables. Further, such methods are not well parallelizable and can take many iterations to converge.

These drawbacks can be overcome by splitting the QUBO into independent subproblems and subsequently merging the solutions into a (hopefully) good solution for the original problem. This approach



(a) QUBO size too large. (b) Divide. (c) Solve and merge. (d) Combine.

Figure 5.1: Overview of our proposed recursive Divide-and-Conquer algorithm for reducing the problem size of a given QUBO instance. A large dataset or feature dimensionality can lead to a huge QUBO matrix size, intractable to be solved on current intermediate-scale quantum hardware (a). The large problem is iteratively split into independent subproblems (divide) until it is solvable with a quantum device (conquer) (b). Since independent subproblems only consider locality, global information is introduced by using current solutions for further subproblem generation (c). Finally, all solutions are combined for obtaining a solution to the original QUBO instance (d).

is known as D&C, where an intractable problem is divided into smaller problems, until they can be solved (conquered). Considering only independent structures leads to a huge information loss for dependencies between the single components. Thus, we propose a recursive D&C algorithm (see Figure 5.1), which iteratively introduces global correlation into the subproblems in Section 5.3.2. It creates new subproblems based on previously obtained solutions, which is especially suited for k-hot constraints, obtaining a trade-off between problem size and number of iterations.

In computer vision, one deals with image data, where each data point (image) can consist of millions of features (pixels). Considering each pixel as a potential candidate in a CO formulation is intractable to be solved with NISQ hardware. Despite these limitations, we explain how a relevant class of computer vision problems can be transferred to the quantum domain. More precisely, we consider a situation in which a set of 3-dimensional points from a scene is viewed by multiple cameras. Given a list of image coordinates of these points in the camera coordinates, finding the set of camera positions, altitudes, imaging parameters, and the point's 3-dimensional locations is a reconstruction process. BA is the estimation that involves the minimization of the re-projection error. It usually goes through an iterative process and requires a good initialization [174]. Solutions to this task allow for the extraction of 3-dimensional coordinates that describe the image geometry and the intrinsic coordinate system of each image. By fusing the available images to construct a single large image, one may eventually extract valuable information, e.g. a classification of each pixel as to whether it belongs to a moving object. In Section 5.4, we investigate the performance of our proposed D&C algorithm in solving a certain subtask of BA. That is, we re-interpret the *keypoint extraction* subtask as a VQ problem that introduces global correlation between all pixels instead of just local ones, which is discussed in Section 5.4.1. For solving this large-scale problem, we adapt our efficient D&C algorithm for incorporating global

information between independent subproblems, where special care is taken for the k hot constraints. Based on the retrieved keypoints, we construct a QUBO formulation for solving a *matching* problem of identifying corresponding keypoint pairs in Section 5.4.2. We replace distance computations using Mercer kernels, circumventing possible drawbacks of feature descriptors. Especially, we investigate the performance of quantum kernels, combining QGC and AQC for the first time. Lastly, in Section 5.5, all proposed methods are evaluated on satellite image data using NISQ devices, such as D-Wave annealers and IBM devices, and DA hardware (quantum emulator).

5.1 Related Work

Iteratively breaking down a large QUBO into subproblems is often used in the literature as well as in state-of-the-art solvers. Starting with an initial guess, such algorithms select an index set and optimize the corresponding sub-QUBO, repeating this process until a fixed number of steps or convergence is achieved [175]. Different strategies exist for selecting subproblems, e. g., by targeting variables promising the highest energy gain [176]. To include constraints, structure-based selection interprets the QUBO matrix as a weighted adjacency matrix of an underlying graph, utilizing methods like community detection [177] and graph clustering [178]. A recursive approach, such as Recursive QAOA (RQAOA) [179] and warm-starting techniques [180], iteratively reduces problem size. Independent subproblem solving, including the divide-and-conquer method [181], can mitigate the need for an initial solution but may sacrifice global solution quality [182].

Considering the field of computer vision, *Quantum Image Processing* (QIP) [183, 184] has emerged as an extensive research field in recent years. The key idea is to encode an image into a quantum state which can then be preprocessed efficiently with quantum routines. Examples include image compression [185, 186], filtering [187], segmentation [188] and edge detection [189–191], but also ML related problems such as nearest neighbors [192] or pattern recognition [193]. For our proposed BA use-case, extracting characteristic keypoints in images is a vital subtask. Obtaining image features is investigated in [194, 195], where interesting pixels are identified locally. Matching multiple images is another important subtask in BA. Using QIP, [196] proposes an image matching method solely on quantum state representations. Having local image features (keypoints) at hand [145, 197] investigate permutation-based QUBO formulations for successful matching.

A caveat of QIP is the encoding of classical images into a quantum state, which is often the bottleneck in the aforementioned methods. Furthermore, these methods only consider extracting local features from images. We propose a recursive D&C algorithm for QUBO formulations built upon classical data which introduces global correlations by recombining solutions of independent subproblems. We also consider a new QUBO formulation for the matching problem, perfectly adapted towards solving BA.

5.2 Incorporating Inequality Constraints

Assume we are posed with a QBP of the following form

$$\min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{W}\boldsymbol{z} \tag{5.1a}$$

$$\text{s.t. } \boldsymbol{A}\boldsymbol{z} \preceq \boldsymbol{b}\,, \tag{5.1b}$$

where $\boldsymbol{W} \in \mathbb{R}^{n\times n}$ characterizes a quadratic CO objective and $\boldsymbol{A} \in \mathbb{Z}^{m\times n}$, $\boldsymbol{b} \in \mathbb{Z}^n$ describe m linear inequality constraints. As we have seen in Section 3.2.1, we can introduce penalty parameters to obtain an equivalent QUBO formulation for Equation (5.1) if we are faced with equality constraints, i. e., $\boldsymbol{Az} = \boldsymbol{b}$. For inequality constraints of the form $\boldsymbol{Az} \preceq \boldsymbol{b}$, the validity of this method does not hold anymore. To ensure equivalence, we can introduce slack variables.

Proposition 5.1. *Let $\boldsymbol{W} \in \mathbb{R}^{n\times n}$, $\boldsymbol{A} \in \mathbb{Z}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{Z}^n$. There exist $\boldsymbol{\lambda} \in \mathbb{R}^m_+$ and $k \in \mathbb{N}$, s.t.*

$$\min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{W}\boldsymbol{z} \quad \text{s.t. } \boldsymbol{Az} \preceq \boldsymbol{b} \quad \Leftrightarrow \quad \min_{\boldsymbol{z}\in\mathbb{B}^n, \boldsymbol{S}\in\mathbb{B}^{k\times m}} \boldsymbol{z}^\top \boldsymbol{W}\boldsymbol{z} + \sum_{i\in[m]} \lambda_i \left(\boldsymbol{A}_i^\top \boldsymbol{z} - b_i + \boldsymbol{p}^\top \boldsymbol{S}_i\right)^2 ,$$

where $\boldsymbol{p} = (2^0, 2^1, \dots, 2^{k-1})$.

We obtain an equivalent QUBO formulation, with $\boldsymbol{P}_1 = (\boldsymbol{I}_n, \boldsymbol{0}_n\boldsymbol{0}_{km}^\top)$, $\boldsymbol{P}_2 = (\boldsymbol{0}_{km}\boldsymbol{0}_n^\top, \boldsymbol{I}_{km})$, $\boldsymbol{\lambda}' := (\sqrt{\lambda_i})_{i\in[m]}^\top$, $\boldsymbol{P} = \boldsymbol{1}_m(\boldsymbol{1}_m \otimes \boldsymbol{p})^\top \in \mathbb{B}^{m\times km}$, $\boldsymbol{A}' := \boldsymbol{\lambda}'\boldsymbol{1}^\top \odot (\boldsymbol{AP}_1 + \boldsymbol{PP}_2)$ denoting row-wise multiplication of $\boldsymbol{\lambda}'$ to $(\boldsymbol{AP}_1 + \boldsymbol{PP}_2)$ and $\boldsymbol{b}' := \boldsymbol{\lambda}' \odot \boldsymbol{b}$

$$\min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{W}\boldsymbol{z} \quad \text{s.t. } \boldsymbol{Az} \preceq \boldsymbol{b} \quad \Leftrightarrow \quad \min_{\boldsymbol{z}\in\mathbb{B}^{n+mk}} \boldsymbol{z}^\top \boldsymbol{Q}\boldsymbol{z}, \quad \boldsymbol{Q} := \boldsymbol{P}_1^\top \boldsymbol{Q}\boldsymbol{P}_1 + \boldsymbol{A}'^\top \boldsymbol{A}' - 2\,\mathrm{diag}(\boldsymbol{b}'^\top \boldsymbol{A}') .$$

Proof. Note that the following equality holds by using an auxiliary vector $\boldsymbol{a} \in \mathbb{N}^m$

$$\min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{W}\boldsymbol{z} \quad \text{s.t. } \boldsymbol{Az} \preceq \boldsymbol{b} \quad \Leftrightarrow \quad \min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{W}\boldsymbol{z} \quad \text{s.t. } \boldsymbol{Az} + \boldsymbol{a} = \boldsymbol{b},\ \boldsymbol{a} \in \mathbb{N}^m .$$

Every natural number $a \in \mathbb{N}$ can be written in binary representation $\boldsymbol{s} \in \mathbb{B}^{k'}$, i. e., $a = s_1 \cdot 2^0 + s_1 \cdot 2^1 + \dots + s_1 \cdot 2^{k'-1} = \boldsymbol{p}^\top \boldsymbol{s}$, where $k' = \lceil \log_2(a) \rceil$ and $\boldsymbol{p} = (2^0, 2^1, \dots, 2^{k'-1})^\top \in \mathbb{N}^{k'}$. Thus, we can represent entry $a_i(\boldsymbol{z}) = b_i - \boldsymbol{A}_i^\top \boldsymbol{z}$ of the auxiliary vector with

$$k_i = \max_{\boldsymbol{z}\mathbb{B}^n} \left\lceil \log_2 \left(b_i - \boldsymbol{A}_i^\top \boldsymbol{z}\right) \right\rceil ,$$

binary variables. Since we have m constraints, we can represent the whole vector $\boldsymbol{a}$ with $k = m \cdot \max_{i\in[m]} k_i$ slack variables, i. e., $\boldsymbol{a} = \boldsymbol{p}^\top \boldsymbol{S}$, $\boldsymbol{S} \in \mathbb{B}^{k\times m}$. Hence, we obtain the equivalence

$$\min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{W}\boldsymbol{z} \quad \text{s.t. } \boldsymbol{Az} \preceq \boldsymbol{b} \quad \Leftrightarrow \quad \min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{W}\boldsymbol{z} \quad \text{s.t. } \boldsymbol{Az} + \boldsymbol{p}^\top \boldsymbol{S} = \boldsymbol{b},\ \boldsymbol{S} \in \mathbb{B}^{k\times m} .$$

With using the argument for equality constraints in Proposition 3.1, we deduce the first claim.

For obtaining an equivalent QUBO formulation, we note that $\boldsymbol{P}_1 = (\boldsymbol{I}_n, \boldsymbol{0}_n\boldsymbol{0}_{km}^\top) \in \mathbb{B}^{n\times n+km}$ projects $\boldsymbol{z} \in \mathbb{B}^{n+km}$ on its first n entries and $\boldsymbol{P}_2 = (\boldsymbol{0}_{km}\boldsymbol{0}_n^\top, \boldsymbol{I}_{km}) \in \mathbb{B}^{km\times n+km}$ on its last km entries. In fact,

Algorithm 3 IterativeSubQUBO

Input: QUBO instance $\boldsymbol{Q} \in \mathcal{Q}_n$
Output: Solution $\boldsymbol{z}' \in \mathbb{B}^n$
1: $\boldsymbol{z}' \leftarrow$InitialSolution($\boldsymbol{Q}$)
2: **while** not converged **do**
3: $\quad \mathcal{I} \leftarrow$ ChooseIndices($\boldsymbol{Q}, \boldsymbol{z}'$)
4: $\quad \boldsymbol{z}' \leftarrow \arg\min_{\boldsymbol{z} \in \mathbb{B}^n_{\mathcal{I} \leftarrow \boldsymbol{z}'_{\mathcal{I}}}} \boldsymbol{z}^\top \boldsymbol{Q} \boldsymbol{z}$
5: **end while**

the following holds true with $\boldsymbol{s} \coloneqq \mathrm{vec}(\boldsymbol{S}) \in \mathbb{B}^{km}$

$$\boldsymbol{a} = \boldsymbol{p}^\top \boldsymbol{S} = \mathbf{1}_m (\mathbf{1}_m \otimes \boldsymbol{p})^\top \boldsymbol{s} = \boldsymbol{P}\boldsymbol{s} \; .$$

Thus we can rewrite

$$\boldsymbol{A}\boldsymbol{z} + \boldsymbol{P}\boldsymbol{s} = \boldsymbol{b}, \; \boldsymbol{z} \in \mathbb{B}^n, \boldsymbol{s} \in \mathbb{B}^{km} \; \Leftrightarrow \; \boldsymbol{A}\boldsymbol{P}_1 \boldsymbol{z}' + \boldsymbol{P}\boldsymbol{P}_2 \boldsymbol{z}' = \boldsymbol{b}, \; \boldsymbol{z}' \in \mathbb{B}^{n+km} \; ,$$

leading to the claimed formulation. □

Proposition 5.1 shows that it is possible to incorporate linear inequality constraints into the quadratic objective, leading to a QUBO formulation. However, this comes with the drawback of expanding the search spice, since we need to introduce mk slack variables with $k \geq 1$. With a large number of constraints m, the resulting QUBO matrix size can be way too large for what current NISQ devices can handle. In practice, one can avoid an exploding QUBO size by using iterative methods such as the *alternating direction method of multipliers* [198], *satellite methods* [199] or *unbalanced penalization* [200].

However, the problem size without incorporating the constraints can already be too large, either through the size of the dataset or the feature dimensionality. Thus, one often considers lower-dimensional subproblems.

5.3 Iteratively Solving Subproblems

We define the notion of a sub-QUBO.

Definition 5.1 (Sub-QUBO)**.** Let $\boldsymbol{Q} \in \mathcal{Q}_n$, $\boldsymbol{z}' \in \mathbb{B}^n$, $\mathcal{I} \subset [n]$ and $\mathcal{I}^c = \mathcal{I} \setminus [n]$. The sub-QUBO of $\boldsymbol{Q}$ w.r.t. $\boldsymbol{z}'$ and $\mathcal{I}$ is defined as

$$\min_{\boldsymbol{z} \in \mathbb{B}^n_{\mathcal{I} \leftarrow \boldsymbol{z}'_{\mathcal{I}}}} \boldsymbol{z}^\top \boldsymbol{Q} \boldsymbol{z} \Leftrightarrow \min_{\boldsymbol{z} \in \mathbb{B}^{n-|\mathcal{I}|}} \boldsymbol{z}^\top \boldsymbol{Q}' \boldsymbol{z}, \quad \boldsymbol{Q}' \coloneqq \boldsymbol{Q}_{\mathcal{I},\mathcal{I}} + \mathrm{diag}\left(\boldsymbol{z}'^\top_{\mathcal{I}} \boldsymbol{Q}_{\mathcal{I},\mathcal{I}^c} + \boldsymbol{Q}_{\mathcal{I}^c,\mathcal{I}} \boldsymbol{z}'_{\mathcal{I}} \right) \; .$$

This definition gives rise to an iterative algorithm [175] shown in Algorithm 3: Starting with some initial first guess solution $\boldsymbol{z}'$ for $\boldsymbol{Q}$ (Algorithm 3), we iteratively choose an index set $\mathcal{I} \subset [n]$ (Algorithm 3) and optimize the corresponding sub-QUBO (Algorithm 3). This process is repeated for a fixed amount of steps our until convergence.

Clearly, we have certain degrees of freedom in the configuration of this algorithm. That is, how do we obtain the initial solution and the indices to optimizer over?

5.3.1 Choosing Subproblems

A straightforward way is to choose the subproblems randomly. However, this can lead to bad solution quality and a lot of iterations for convergence.

Energy-Based One can also use more informative heuristic methods, for example optimizing the variables which promise the largest energy gain. This is a popular tool and is implemented in powerful hybrid solvers from D-Wave, such as *QBSolv* or *Kerberos*[1]. The possible *energy gain* for a single variable z'_i for a given assignment $\boldsymbol{z}' \in \mathbb{B}^n$ is given by

$$E^i_{\boldsymbol{Q}}(\boldsymbol{z}') := (1 - 2z'_i)\left(Q_{ii} + \sum_{j \in [m] \setminus \{i\}} (Q_{ij} + Q_{ji}) z'_j\right) ,$$

The index set $\mathcal{I}$ can then for example be chosen as the variables with the best k variables or cutting off at a certain energy threshold. Using this method, an immediate threat is to end up in a local optimum of the QUBO energy landscape.

Structure-Based Since QUBO matrices are quadratic, they can also be interpreted as a weighted adjacency matrix of an underlying graph. Every structural element (connectivity, number of edges, etc.) is fully contained in $\boldsymbol{Q}$. Thus, it may be beneficial to search for natural groups of variables, when solving subproblems. Popular methods include searching for connected components, *community detection* [177] or graph clustering methods [178], such as *spectral clustering* [201].

In practice, CO problems are not directly available in QUBO form, but rather incorporate constraints, such as QBP. This gives rise to another decomposition method, which groups variables by given constraints. However, if the number of constraints is very large or effect many variables simultaneously, this method might not lead to a large problem size reduction.

Recursive Recursion is another interesting approach for iteratively solving subproblems. This is done by recursively reducing the problem size with an already available solution. It has been adapted to QAOA [179], called recursive QAOA (RQAOA), and expanded to possible *warm-starting* in [180]. For an overview of QAOA, we refer to Chapter 2. Further, state-of-the-art algorithms for integer optimization are often given in the *Branch-and-Bound* (B&B) framework. For QUBO, one can relax the optimization over binary variables to the interval $[0, 1]$. Subsequent branching to obtain binary variables, leads to a recursive algorithm for QUBO [202] with reduced problem size.

5.3.2 Recursive Divide-and-Conquer

Iteratively solving subproblems comes with some caveats. If the problem size is very large, we often need a lot of iterations for Algorithm 3 to converge or for finding a solution with a satisfactory quality. Due to the purely iterative nature, it is not directly possible to exploit independence in the subproblems.

[1] https://docs.ocean.dwavesys.com/en/stable/docs_hybrid/reference/reference.html (last accessed April 8, 2025)

Algorithm 4 INDEPENDENTSUBQUBO

Input: QUBO instance $\boldsymbol{Q} \in \mathcal{Q}_n$
Output: Solution $\boldsymbol{z}' \in \mathbb{B}^n$
1: $I \leftarrow$DIVIDEINDICES($\boldsymbol{Q}$)
2: $\mathcal{Z} \leftarrow \emptyset$
3: **for** $\mathcal{I}$ in I **do**
4: $\quad \boldsymbol{z}' \leftarrow \arg\min_{\boldsymbol{z} \in \mathbb{B}^n_{\mathcal{I}^c \leftarrow \mathbf{0}}} \boldsymbol{z}^\top \boldsymbol{Q} \boldsymbol{z}$
5: $\quad \mathcal{Z} \leftarrow \mathcal{Z} \cup \{\boldsymbol{z}'\}$
6: **end for**
7: $\boldsymbol{z}' \leftarrow$ MERGESOLUTIONS($\mathcal{Z}$)

Algorithm 5 RECURSIVESUBQUBO

Input: QUBO instance $\boldsymbol{Q} \in \mathcal{Q}_n$ and $\mathcal{J} \subset [n]$
Output: Solution $\boldsymbol{z}' \in \mathbb{B}^n$
1: $I \leftarrow$DIVIDEINDICES($\boldsymbol{Q}, \mathcal{J}$)
2: $J \leftarrow \emptyset$
3: **for** $\mathcal{I}$ in I **do**
4: $\quad \boldsymbol{z} \leftarrow$ RECURSIVESUBQUBO($\boldsymbol{Q}, \mathcal{I}$)
5: $\quad J \leftarrow J \cup \{j : z_j = 1, j \in [n]\}$
6: **end for**
7: $\boldsymbol{z}' \leftarrow \arg\min_{\boldsymbol{z} \in \mathbb{B}^n_{J^c \leftarrow \mathbf{0}}} \boldsymbol{z}^\top \boldsymbol{Q} \boldsymbol{z}$

Another approach is to focus on disjoint index sets and solve problems independently. Independence of an index set $\mathcal{I}$ to all other indices $\mathcal{I}^c$ means, that we assume all indices in $\mathcal{I}^c$ to take the value 0 in the sub-QUBO sense. That is, no QUBO matrix entry of the rows or columns in $\mathcal{I}^c$ effects the solution of the subproblem

$$\min_{\boldsymbol{z} \in \mathbb{B}^{|\mathcal{I}|}} \boldsymbol{z}^\top \boldsymbol{Q}_{|\mathcal{I}|,|\mathcal{I}|} \boldsymbol{z} \Leftrightarrow \min_{\boldsymbol{z} \in \mathbb{B}^n_{\mathcal{I}^c \leftarrow \mathbf{0}}} \boldsymbol{z}^\top \boldsymbol{Q} \boldsymbol{z} \,.$$

An algorithm for breaking a QUBO into independent instances is given in Algorithm 4. Even though we have no need for an initial solution anymore, we are given independent solutions to each subproblem. The obvious way of simply setting the subvectors to the corresponding solution may not lead to useful results, since the resulting configuration might not respect all given constraints. One thus has to carefully design a merging mechanism (Algorithm 4) for obtaining a feasible solution to the original problem.

The procedure of dividing a problem into independently solvable subproblems and subsequent merging of the solutions is known as *divide-and-conquer* (D&C). It has been first applied to QUBO in [181] and is often combined with a recursive approach to reduce the corresponding problem size [182]. However, considering only independent subproblems leads to crucial information loss. Even though the subproblems might be solved to optimality, we have no guarantees on the global quality of the solution. Thus, we propose a D&C algorithm which iteratively solves independent subproblems and induces global information. An algorithmic overview is given in Algorithm 5. As we can see from Algorithm 5, the new subproblems are constructed from the already obtained solutions.

Depending on the given constraints and the underlying problem structure, this method can be very

suitable. Specifically, consider QBP problems of the following form

$$\min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{W}\boldsymbol{z} \tag{5.2a}$$

$$\text{s.t. } \mathbf{1}^\top \boldsymbol{z} = k\ , \tag{5.2b}$$

where $k \leq n$. Following Section 3.2.1, we can introduce a penalty to obtain an equivalent QUBO formulation.

In Equation (5.2), we want to find the minimizing k-hot vector. Since an optimal solution always contains exactly k non-zero entries, Algorithm 5 is guaranteed to add the same number of indices in every step. This leads to a well controllable problem size. In fact, for $k = n$, we exactly recover the original formulation. Thus, there is a trade-off between the problem size and the number of used recursive iterations. An application of the recursive D&C algorithm is given in the next section.

5.4 Application: Bundle Adjustment

We consider a scenario in which a set of three-dimensional points from a scene is observed by V cameras. Given the image coordinates of these points in the respective camera coordinate systems, the task of determining the camera positions, orientations, imaging parameters, and the three-dimensional locations of the points constitutes a reconstruction process. A crucial step in this process is BA, which aims to refine these estimates by minimizing the re-projection error—the discrepancy between the observed image points and their predicted projections based on the current model parameters. This optimization procedure is typically iterative and requires a well-chosen initialization to ensure convergence to an accurate solution [174].

Solutions to this task enable the extraction of three-dimensional coordinates that describe both the image geometry and the intrinsic coordinate system of each image. By merging multiple images into a single, unified representation, valuable insights can be obtained—such as determining whether individual pixels correspond to moving objects. This process can be divided into several key subtasks:

1. **Keypoint Extraction**: The first step involves identifying keypoints (also known as interest points or feature points) in each image. These keypoints correspond to distinctive pixels, such as corner points. In a naive approach, every pixel could serve as a keypoint, but this is typically computationally expensive.

2. **Keypoint Matching**: Once keypoints are extracted, those shared across multiple images must be matched. Given the potentially large number of pixels, both keypoint extraction and matching are computationally intensive tasks.

3. **Coordinate System Alignment**: Using the correspondences between matched keypoints, a projection is determined that aligns the coordinate systems of all images.

4. **Image Fusion and Object Segmentation**: After aligning the images, they are fused into a single composite image. In this step, missing regions are identified, and overlapping areas with conflicting pixel data are analyzed to detect and segment moving objects.

The final alignment step is done by finding transformations between different images which align them to a single plane. These transformations can have varying forms, e.g. a homography or a fundamental

matrix. Without further processing, this subtasks is inherently continuous and thus not very well suited for quantum computation. Classical methods include the "eight-point algorithm" [203], direct linear transformation (DLT) [204], and enhanced correlation coefficient (ECC) [205]. The refinement of the alignment step is referred to as BA [34].

Mathematically, the problem can be formulated as follows: Assume that P 3-dimensional points are visible through V different views and let $\boldsymbol{\pi}_{ij}$ be the projection of the i-th point onto the plane containing the j-th image. Since $\boldsymbol{\pi}_{ij}$ may not lie in the image itself, we define a binary variable b_{ij} which is 1 if and only if point i is visible in image j. Furthermore, assume that the camera that created the j-th image can be characterized by a vector $\boldsymbol{w}_j$, and every 3-dimensional point i by a vector $\boldsymbol{r}_i$. The objective is now to minimize the total re-projection error

$$f_{BA}(\boldsymbol{w}_j, \boldsymbol{r}_i) = \sum_{i=1}^{P} \sum_{j=1}^{V} b_{ij} \left\| \tilde{\pi}(\boldsymbol{w}_j, \boldsymbol{r}_i) - \boldsymbol{\pi}_{ij} \right\|_2^2 , \tag{5.3}$$

where $\tilde{\pi}$ corresponds to the predicted projection.

The goal of this section is to investigate how far current QC resources can be used to tackle the problem of BA. As explained above, we focus on keypoint extraction and feature matching, which are both computationally hard problems. Due to their discrete structure, these subtasks exhibit a large potential for improvements via quantum computation, as known from other areas of signal processing [206].

In general, pixel data can either be represented by raw color channels, or sophisticated feature space mappings, e.g., scale-invariant feature transform (SIFT) [207], (accelerated) KAZE [208, 209], or low-dimensional embeddings based on geometric hashing [205, 210]. Nevertheless, if not stated differently, we assume that an image is encoded as a set of pixel features $\mathcal{I} = \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_M\}$. For example, $\boldsymbol{p_i} = (x_i, y_i, r_i, g_i, b_i)$ represents a pixel with location (x_i, y_i) and color channels (r_i, g_i, b_i).

5.4.1 Keypoint Extraction

The goal of keypoint detection is to extract a subset of relevant pixels which describe the full image well—we re-interpret this step as a VQ problem. Due to the NP-hardness of VQ, offloading the corresponding computation to a quantum processor promises large benefits. We recall the VQ problem settings from Chapter 2 and the corresponding QUBO formulations from Chapter 3. Specifically, the VQ objective respective the k-medoids objective (KMEDVQ) is given by

$$\min_{\boldsymbol{z} \in \mathbb{B}^n} \; -\alpha \boldsymbol{z}^\top \boldsymbol{D} \boldsymbol{z} + \beta \left(\boldsymbol{D} \boldsymbol{1} \right)^\top \boldsymbol{z} \tag{5.4a}$$

$$\text{s.t. } \boldsymbol{1}^\top \boldsymbol{z} = k \, , \tag{5.4b}$$

where $\boldsymbol{D}$ is a distance matrix and $\alpha, \beta > 0$ are parameters for balancing the two objectives of finding central and far apart prototypes. As we have seen in Proposition 3.6, an equivalent formulation for specific α, β and $\boldsymbol{D}$ is given by mean discrepancy VQ (MDVQ)

$$\min_{\boldsymbol{z} \in \mathbb{B}^n} \; \frac{1}{k^2} \boldsymbol{z}^\top \boldsymbol{K} \boldsymbol{z} - \frac{2}{nk} (\boldsymbol{K} \boldsymbol{1})^\top \boldsymbol{z} \tag{5.5a}$$

$$\text{s.t. } \boldsymbol{1}^\top \boldsymbol{z} = k \, , \tag{5.5b}$$

where $\boldsymbol{K}$ is the kernel matrix of an underlying Mercer kernel. Both QBP formulations can be brought into QUBO form by integrating the constraint with the help of a penalty parameter $\lambda > 0$ (Proposition 3.1)

$$\min_{\boldsymbol{z}\in\mathbb{B}^n} \boldsymbol{z}^\top \boldsymbol{W}\boldsymbol{z} \quad \Leftrightarrow \quad \boldsymbol{z}^\top \boldsymbol{W}\boldsymbol{z} + \lambda\left(\boldsymbol{z}^\top \mathbf{1}\mathbf{1}^\top \boldsymbol{z} - 2k\mathbf{1}^\top \boldsymbol{z}\right) . \tag{5.6a}$$

$$\text{s.t. } \mathbf{1}^\top \boldsymbol{z} = k \tag{5.6b}$$

Up to now, classical Gaussian kernels, with $K(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\gamma\|\boldsymbol{x} - \boldsymbol{y}\|_2^2)$, have been considered for KDC in the literature. Here, however, we also consider quantum kernels, i.e. the kernel matrix $\boldsymbol{\mathcal{K}}$ is computed via the quantum gate circuit that is shown in Figure 2.4.

Depending on the specific choice of the parameters α and β in KMEDVQ, MDVQ results in prototypes that contain more information for dense parts of an image and hence deliver substantially different results. Both QBP formulations given in Equations (5.4) and (5.5) exactly have the form of optimizing over k-hot vectors Equation (5.2), so we can apply our recursive D&C algorithm (Algorithm 5) for reducing the problem size.

5.4.2 Feature Matching

Based on the output of keypoint extraction, suiting matches between keypoints of different images must be identified. To this end, let $\boldsymbol{p}_i^{\mathcal{I}_1}, 1 \leq i \leq n$ and $\boldsymbol{p}_j^{\mathcal{I}_2}, 1 \leq j \leq m$ be keypoints extracted from images $\mathcal{I}_1$ and $\mathcal{I}_2$, respectively. The task is to find pairs $(\boldsymbol{p}_i^{\mathcal{I}_1}, \boldsymbol{p}_j^{\mathcal{I}_2})$ s.t. $\boldsymbol{p}_i^{\mathcal{I}_1}$ in the first image, corresponds to $\boldsymbol{p}_j^{\mathcal{I}_2}$ in the second image. For this task, raw information about a single pixel is insufficient, since both images might be scaled, rotated or illuminated in a different way. Thus, feature descriptors, e.g. SIFT or AKAZE, are computed for each keypoint, i.e. $\boldsymbol{x}_i$ for $\boldsymbol{p}_i^{\mathcal{I}_1}$ and $\boldsymbol{y}_j$ for $\boldsymbol{p}_j^{\mathcal{I}_2}$. Let f be the function that maps keypoints $\boldsymbol{p}_i^{(1)}$ and $\boldsymbol{p}_j^{(2)}$ to their feature descriptors $\boldsymbol{x}_i = f(\boldsymbol{p}_i^{(1)})$ and $\boldsymbol{y}_j = f(\boldsymbol{p}_j^{(2)})$. The distance between feature descriptors of different keypoints $\left\|\boldsymbol{x}_i - \boldsymbol{y}_j\right\|_2$ can then be used to measure the similarity.

As a first step towards obtaining a matching, we identify each $\boldsymbol{x}_i$ with its k nearest neighbors $\{\boldsymbol{y}_{j_1}, \ldots, \boldsymbol{y}_{j_k}\}$, since it can happen that a keypoint is very similar to a non-corresponding point. In practice, one often chooses k fairly small, e.g., $k = 2$. We formulate this problem as a QUBO, since matching is known to be NP-hard. Since this procedure allows for identifying possible mismatches, typically a classical postprocessing method is used, such as *RANSAC* [211].

Proposition 5.2. *Let $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}, \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m\} \subset \mathbb{R}^d$, $D : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ a distance function with corresponding distance matrix $\boldsymbol{D} \in \mathbb{R}^{n\times m}$. Further, let $\boldsymbol{p} = (2^0, 2^1, \ldots, 2^{k-1})$, $k > 0$, $l = \lceil \log_2(k) \rceil$, $\boldsymbol{P}_1 = (\boldsymbol{I}_{nm}, \boldsymbol{0}_{nm}\boldsymbol{0}_{nl}^\top)$, $\boldsymbol{P}_2 = (\boldsymbol{0}_{nl}\boldsymbol{0}_{nm}^\top, \boldsymbol{I}_{nl})$, $\boldsymbol{P} = \mathbf{1}_n(\mathbf{1}_n \otimes \boldsymbol{p})^\top \in \mathbb{B}^{n\times nl}$, $\boldsymbol{B} = \mathbf{1}_n^\top \otimes (\mathbf{1}_n \otimes \boldsymbol{I}_m)$ and $\boldsymbol{A} = \boldsymbol{I}_n \otimes \mathbf{1}_m^\top$ and $\boldsymbol{A}' := (\boldsymbol{A}\boldsymbol{P}_1 + \boldsymbol{P}\boldsymbol{P}_2)$. A QUBO formulation for the matching problem is given by*

$$\min_{\boldsymbol{z}\in\mathbb{B}^{n(m+l)}} \boldsymbol{z}^\top \boldsymbol{Q}\boldsymbol{z}, \tag{5.7a}$$

$$\boldsymbol{Q} := \beta \boldsymbol{P}_1^\top \boldsymbol{B}\boldsymbol{P}_1 + \gamma \boldsymbol{A}'^\top \boldsymbol{A}' + \operatorname{diag}\left(\left(\operatorname{vec}(\boldsymbol{D}) - \alpha\mathbf{1}_{mn}\right)\boldsymbol{P}_1 - 2\gamma k \mathbf{1}_{nm}^\top \boldsymbol{A}'\right) , \tag{5.7b}$$

where α, β, γ are penalty parameters, weighing the different objectives.

Proof. The optimization objective for the matching problem can be formulated as a QBP problem by

indicating a potential match between points $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ by a binary variable $Z_{ij} \in \mathbb{B}$. We vectorize the corresponding binary matrix $\boldsymbol{z} = \mathrm{vec}(\boldsymbol{Z})$ to obtain

$$\min_{\boldsymbol{z}\in\{0,1\}^{nm}} \quad \boldsymbol{z}^\top \mathrm{vec}\,(\boldsymbol{D}) \tag{5.8a}$$

$$\text{and} \quad \max_{\boldsymbol{z}\in\{0,1\}^{nm}} \quad \boldsymbol{z}^\top \mathbf{1}_{nm} \tag{5.8b}$$

$$\text{subject to} \quad \boldsymbol{z}^\top \boldsymbol{B}\boldsymbol{z} = \mathbf{1}_{nm}^\top \boldsymbol{z} \tag{5.8c}$$

$$\boldsymbol{A}\boldsymbol{z} \preceq k\mathbf{1}_n\ , \tag{5.8d}$$

with $\boldsymbol{N} = \mathbf{1}_n^\top \otimes (\mathbf{1}_n \otimes \boldsymbol{I}_m)$ and $\boldsymbol{M} = \boldsymbol{I}_n \otimes \mathbf{1}_m^\top$. We are faced with two objectives: minimizing the distances between matched points Equation (5.8a) and maximizing the number of matches Equation (5.8b). The condition in Equation (5.8c) ensures that two points $\boldsymbol{x}_{i_1}$ and $\boldsymbol{x}_{i_2}$ are not matched to the same point $\boldsymbol{y}_j$ while Equation (5.8d) forces every point $\boldsymbol{x}_i$ to be matched with maximally k points $\boldsymbol{y}_{j_1}, \ldots, \boldsymbol{y}_{j_k}$. Since $\boldsymbol{z}^\top \boldsymbol{S}\boldsymbol{z} \geq \mathbf{1}_{nm}^\top \boldsymbol{z}$, condition Equation (5.8c) is fulfilled if at its minimum. Thus, we can obtain an equivalent QBP by using penalty factors $\alpha, \beta > 0$

$$\min_{\boldsymbol{z}\in\{0,1\}^{nm}} \quad \boldsymbol{z}^\top \mathrm{vec}\,(\boldsymbol{D}) - \alpha \boldsymbol{z}^\top \mathbf{1}_{nm} + \beta \boldsymbol{z}^\top \boldsymbol{N}\boldsymbol{z} \tag{5.9a}$$

$$\boldsymbol{M}\boldsymbol{z} \preceq k\mathbf{1}_n\ . \tag{5.9b}$$

The inequality constraint in Equation (5.8d) can be reformulated by using a penalty parameters $\gamma > 0$, binary slack variables (Proposition 5.1) and projection matrices $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$, leading to the claimed result. □

The parameters β and γ are chosen to be large enough such that the conditions in Equation (5.8c) and Equation (5.8d) are adhered. Without loss of generality, we assume $D(\boldsymbol{x}_i, \boldsymbol{y}_j) \leq 1$ and choose α to be in $[0, 1]$. A large α emphasizes the maximization of the number of matches in Equation (5.8b) which forces every $\boldsymbol{x}_i$ to be matched with a $\boldsymbol{y}_j$ even though they may not be very similar. By setting α close to 0, the distance minimization in Equation (5.8a) is prioritized—leading to no matches at all in the extreme case. As we have seen in Proposition 3.7, we can define a distance measure using specific Mercer kernels, which leads to the applicability of the kernel trick.

5.4.3 Quantum Kernel Methods

Quantum kernels are special Mercer kernels because they leverage quantum feature maps to encode classical data into a high-dimensional Hilbert space, which is exponentially large compared to classical feature spaces. This allows quantum kernels to potentially provide a quantum advantage by enabling complex feature mappings that are infeasible for classical methods. We recall from Chapter 2 that unlike classical kernels, which use predefined mathematical functions to compute inner products in a feature space, quantum kernels rely on quantum circuits to map data into quantum states. The kernel is then defined as the fidelity (overlap) between these quantum states

$$K(\boldsymbol{x}, \boldsymbol{x}') = \left|\langle \psi(\boldsymbol{x}) | \psi(\boldsymbol{x}')\rangle\right|^2\ ,$$

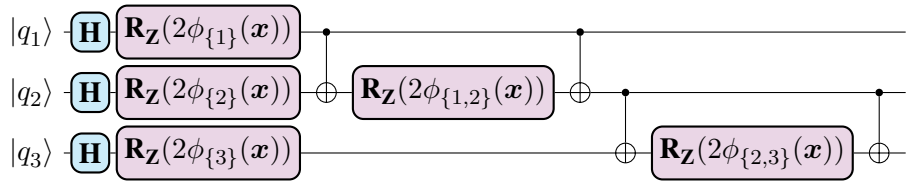


Figure 5.2: Exemplary feature map for three qubits. The *ZZFeatureMap* is depicted, that is $\boldsymbol{U}_{\phi(\boldsymbol{x})}$ Equation (5.10) with $\phi_{\{i\}}(\boldsymbol{x}) = x_i$ and $\phi_{\{i,j\}}(\boldsymbol{x}) = (\pi - x_i)(\pi - x_j)$.

where $|\psi(\boldsymbol{x})\rangle = \mathcal{U}_{\phi(\boldsymbol{x})} |0\rangle^{\otimes d}$ is a quantum state encoding the classical data $\boldsymbol{x}$. The dimension of the Hilbert space grows exponentially with the number of qubits, allowing for intricate, highly nonlinear relationships between data points. One key advantage of quantum kernels is that they implicitly compute similarities in feature spaces that may be classically intractable. Certain quantum feature maps create highly entangled states that are difficult to approximate using classical computations.

To obtain the quantum feature map, the unitary $\mathcal{U}_{\phi(\boldsymbol{x})}$ is implemented by iteratively applying universal unitaries $\boldsymbol{U}_{\phi(\boldsymbol{x})}$. In the literature (e. g. in [99]), it is often suggested to use a Pauli-gate efficient ansatz, i. e.,

$$\boldsymbol{U}_{\phi(\boldsymbol{x})} = \exp\left(-i \sum_{\mathcal{S}\subset[d]} \phi_{\mathcal{S}}(\boldsymbol{x}) \prod_{k\in\mathcal{S}} \mathbf{Z}_k\right) \mathbf{H}^{\otimes d}\,, \tag{5.10}$$

For compatibility with NISQ-devices, some typical pitfalls must be avoided: Considering local feature functions for all subsets $S \subset [N]$ is too costly: To see this, one has to consider the transpilation of user specified quantum circuits. Transpilation is the process of rewriting a given input circuit to match the connectivity structure and noise properties of a specific quantum processor. Most circuits must undergo a series of transformations that make them compatible with a given target device, and optimize them to reduce the effects of noise of the resulting outcomes. Rewriting quantum circuits to match hardware constraints and optimizing for performance can be far from trivial. The flow of logic in the rewriting tool chain need not be linear, and can often have iterative subloops, conditional branches, and other complex behaviors. Most importantly, it encompasses the decomposition of gates involving three or more qubits into 2-qubit gates. Clearly, the heavy hexagon structure is pairwise and hence contains no connection between three or more qubits. As a direct result, an apparently "shallow" quantum gate circuit, consisting of a single unitary operation among N-qubits, can thus eventually exhibit a very high depth. High circuit depths require large decoherence and dissipation times, which are not available in the current generation of NISQ-devices.

It is hence recommended to consider only pairwise features ($S \subset [N] \wedge |S| = 2$) in Equation (5.10) as it is depcited in Figure 5.2. First, a Hadamard gate $\mathbf{H}$ is applied to every qubit to obtain an equal superposition state. Then, the data is encoded into every qubit independently with the help of single-qubit $\mathbf{R_Z}$ gates. Finally, entanglement is created with the help of $\mathbf{R_{ZZ}} = \mathbf{CX}(\boldsymbol{I}_2 \otimes \mathbf{R_Z})\mathbf{CX}$ gates applied to qubits 1 and 2, and 2 and 3, respectively.

Due to the normalization of quantum states ($\langle\psi|\psi\rangle = 1$), we can apply Proposition 3.7 for obtaining a valid distance measure using a quantum kernel. This can be readily used in the distance-based QUBO formulations developed in the previous sections.

5.5 Experimental Evaluation

For our experimental evaluation, we consider 344 sets of $k = 5$ images from the Kaggle "Draper Satellite Image Chronology" challenge[2] which all have a resolution of 3099×2329 pixels.

5.5.1 Experimental Protocol

Each pixel $\boldsymbol{p}$ is represented by a 5-dimensional vector which captures the position in the image as well as the RGB color channels, $\boldsymbol{p} = (x, y, r, g, b)$. We further down-weight the location information by a factor of $1/4$ to emphasize the importance of the color channels. Finally, pixel vectors are normalized, i.e. $\|\boldsymbol{p}\|_2 = 1$.

The raw image resolution implies that the QBPs from Equations (5.4) and (5.5) are $7\,217\,571$-dimensional—far beyond the capabilities of any quantum annealer or gate-based quantum computer. We hence use our D&C algorithm and split the task recursively into subtasks. First, redundant information is reduced by down-sampling images to 928×704 pixels. Then, each image is split into 32×32 equally sized non-overlapping subimages, which results in patches of size 29×22 pixels. Figure 5.3(a) depicts one exemplary patch of the image from Figure 5.3(d). The keypoint extraction on the original image is an iterative process of finding keypoints in the current "layer" and then merging them to form the next dataset. More precisely, 10 keypoints are extracted on every of the 32×32 image patches. The keypoints of adjacent 4×4 image patches are chosen to be the next dataset for VQ. On these datasets 20 keypoints are extracted, which are again grouped to form 4×4 grids. The last step is then to extract 45 keypoints on every of the remaining 4 datasets to obtain 180 final keypoints. The penalty parameters are chosen such that every summand in has approximately the same contribution. For KMEDVQ we set $\alpha = 1/k$, $\beta = 1/n$ and $\lambda = 1/k$, and for MDVQ we set $\lambda = 1/k^2$. Since the parameter λ weighs the constraint of finding exactly k prototypes, setting these values too low can result in finding states not adhering this constraint. For the matching problem we employ SIFT feature descriptors in a normalized inner product kernel. The QUBO parameters (see Proposition 5.2) are set to $k = 1$, $\beta = 1$, $\gamma = 1$, while α is varied for showing the effect of this parameter.

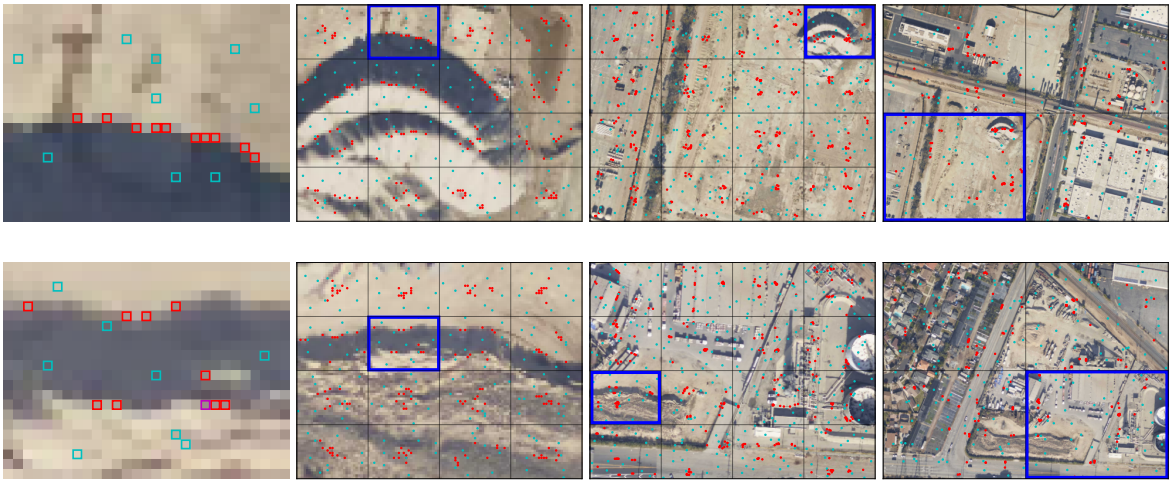
QUBO solvers process the same problem multiple times to prevent local optima. The state with the lowest energy is then chosen to be the solution. We consider a digital annealer (10 shots, runtime $1s$ per shot), a D-Wave Advantage System 5.1 with 5619 qubits (1024 shots, runtime $50\,\mu s$ per shot), SA, and tabu search, the latter being implemented in the D-Wave Ocean SDK[3].

For computing quantum kernels, we consider a statevector simulation and an IBM Falcon superconducting quantum processor with 5 qubits. The circuits on the IBM system are executed with $10\,000$ shots.

5.5.2 Results

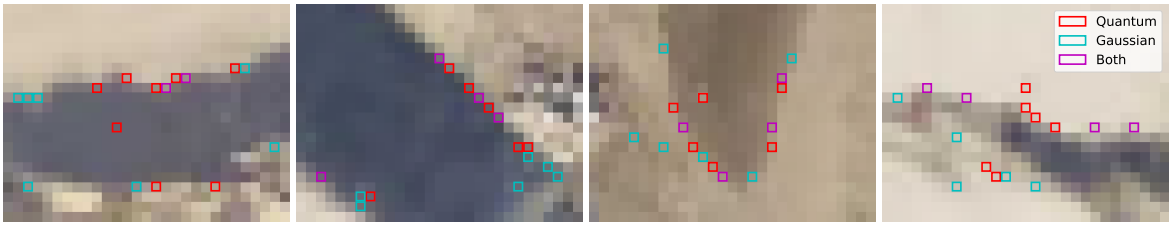
Two exemplary results of the keypoint extraction pipeline can be found in Figure 5.3. It is evident from every single subfigure that KMEDVQ and MDVQ allocate the prototypes substantially different. While KMEDVQ spreads its centroids equally distributed over the whole image patch, MDVQ is able to detect edges which is very useful for keypoint extraction. However, if such edges are represented by only a few pixels in the image patch (low density), MDVQ may not detect them. This is e.g. evident from the left

[2] https://www.kaggle.com/c/draper-satellite-image-chronology/data (last accessed September 19, 2025)
[3] https://docs.ocean.dwavesys.com/en/stable (last accessed September 19, 2025)



(a) Extraction of 10 keypoints on a single image patch. (b) Extraction of 10 keypoints on every image patch. (c) Extraction of 20 keypoints on 4×4 grid. (d) Extraction of 45 (final) keypoints on 4×4 grid.

Figure 5.3: Keypoint extraction is done by subsequently computing prototype pixels on subimages by solving the QUBOs given in Section 5.4.1 using the digital annealers. The image is first split into 32×32 image patches of size 29×22 pixels and ten keypoints are extracted on every patch. The resulting prototypes are then grouped into 8×8 grids of size 4×4 to obtain keypoint sets of size 160 shown in (a) and (b). We extract 20 keypoints on each of these sets and group the resulting prototypes into 2×2 grids of size 4×4 to obtain keypoint sets of size 320 (c). Lastly, 45 prototypes are computed on these sets to obtain the final set of 180 keypoints (d). Results for KMEDVQ are depicted in red, MDVQ in light blue and coincident keypoints in purple.



(a) (b) (c) (d)

Figure 5.4: Extraction of 10 keypoints on 29×22 patches with MDVQ: Comparison between Gaussian and quantum kernel.

Table 5.1: Keypoint extraction on ten 8×8 patches: Comparison of energy values between the D-Wave quantum annealer Advantage System 5.1 (QA), simulated annealing (SA) and digital annealing (DA). Lower is better.

	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)
QA	−4.71	−4.80	−4.74	−4.69	−4.75	−4.34	−4.29	−4.40	−4.44	−4.40
SA	−4.74	−4.82	−4.77	−4.75	−4.78	−4.79	−4.73	−4.78	−4.81	−4.79
DA	**−4.75**	**−4.82**	**−4.77**	**−4.76**	**−4.78**	**−4.79**	**−4.74**	**−4.79**	**−4.81**	**−4.80**

and right neighbor patches of the highlighted patch in the top row of Figure 5.4(c). In such cases, the prototypes are driven towards the center of the patch, since most density is then captured in the position

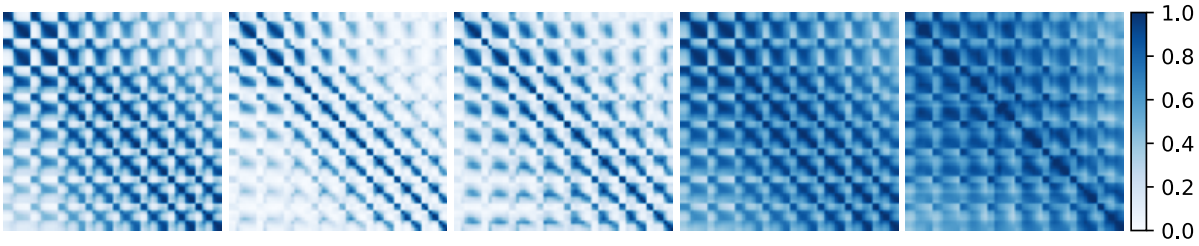


Figure 5.5: Comparison of kernel matrices computed on an exemplary 8×8 image patch shown in Figure 5.6(e). Specifics on the used kernel from left to right: Gaussian kernel, quantum kernel computed with simulation, quantum kernel computed with real quantum hardware, quantum kernel computed with simulation with inputs being scaled by a factor of $s = 0.5$, quantum kernel computed with real quantum hardware with inputs being scaled by a factor of $s = 0.5$.

(a) (b) (c) (d) (e)

Figure 5.6: Extraction of 10 keypoints on 8×8 patches: Comparison of D-Wave quantum annealer and digital annealer.

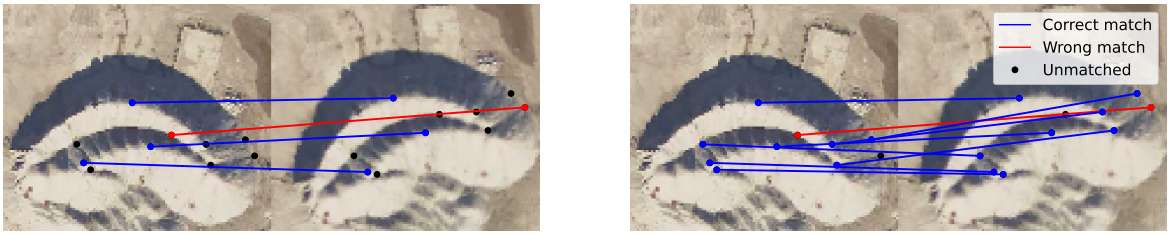
of the pixels—a proper re-weighting of pixel locations can hence be considered as a hyper parameter of the proposed method. In almost all cases, the digital annealer outperforms SA and tabu search.

A comparison between the usage of a Gaussian kernel and a quantum kernel for MDVQ is depicted in Figure 5.4. Keypoints are extracted on four different 29×22 image patches solving the MDVQ QUBO problem with the digital annealer. The quantum kernel is computed via Schrödinger wave-function / statevector simulation—we can see that MDVQ with a quantum kernel distributes its prototypes slightly different to the ones using a Gaussian kernel, while also capturing interesting pixel locations. Constructing a full quantum pipeline can hence be a viable approach.

Figure 5.5 depicts a comparison of kernel matrices of a Gaussian kernel with a quantum kernel computed from the 8×8 patch in Figure 5.6(e). We here show the effects on the kernel matrices of scaling the inputs $\boldsymbol{x} \leftarrow s\boldsymbol{x}$. The quantum kernel matrices are computed for two different scales, $s = 1$ and $s = 0.5$. We compare the results from the statevector simulation with the estimated kernel values using actual quantum hardware. One can see that the quantum kernels computed on actual hardware have a very similar structure to the simulated ones, while the scaling of the inputs substantially affects the "density" of the kernel matrix.

In Figure 5.6 we compare the performance in solving the MDVQ QUBO problem with a quantum annealer and the digital annealer. Five 8×8 image patches are depicted with the corresponding extracted keypoints. Table 5.1 shows the corresponding energy values of the best computed solution. It is clear that the digital annealer is finding better states in terms of objective function value.

Finally, exemplary solutions of the matching QUBO are depicted in Figure 5.7. For this, a subimage



(a) Emphasis on good matches ($\alpha = 0.05$).

(b) Emphasis on many matches ($\alpha = 0.2$).

Figure 5.7: Matching of 10 keypoints on a small image excerpt solving the matching QUBO in Proposition 5.2 with the digital annealer. The right image excerpt corresponds to the left one rotated clockwise by 20°.

with 10 keypoints is rotated by 20° to obtain the same scene from a different view. The keypoints are then matched by solving the QUBO from Proposition 5.2. In this case, the digital annealer needs a larger annealing time ($60s$) to find a good state, while tabu search can identify a good solution rather quickly. This shows that not only the QUBO dimension but especially the underlying energy landscape is of great importance for the performance of finding good states. We can see that setting α to a small value leads to finding only a few matches. However, the identified matches have the highest quality, i.e., the largest kernel values. The wrongly matched pair has a larger kernel value than the theoretically correct match, which can be ascribed to the feature representation of SIFT and is not an artefact of the underlying QUBO.

5.6 Conclusion

In this chapter, we have addressed the challenge of solving large-scale QUBO problems on NISQ devices by developing an efficient recursive D&C algorithm. This method enables the decomposition of complex problems into independent subproblems while iteratively introducing global correlations, thus achieving a balance between problem size reduction and solution accuracy.

We demonstrated the applicability of our approach to computer vision tasks, particularly in the context of BA. By formulating keypoint extraction as a VQ problem, we introduced global correlations beyond local feature descriptors. Furthermore, we constructed a QUBO formulation for the keypoint matching problem, leveraging Mercer kernels to enhance performance and mitigate the limitations of classical feature descriptors. In doing so, we explored the integration of quantum kernels, combining QGC and AQC for the first time. Our methods were empirically evaluated on satellite image data using state-of-the-art quantum and quantum-inspired hardware, including D-Wave annealers, IBM quantum devices, and FPGA-based DA hardware.

Future work includes the optimization of hyper parameters (e.g., penalty parameters) and investigating the "qubitization" of the full BA task, e.g., formulating (5.3) as a QUBO or quantum circuit. Comparing our approaches with classical counterparts and looking at postprocessing methods would also be intriguing. Moreover, quantum kernels can be used in the matching QUBO as well. Respecting the limitations of current quantum hardware, a lower dimensional feature descriptor than SIFT can be chosen, e.g. PCA-SIFT [212]. In any case, our contributed methods open up opportunities for BA and other computer vision tasks on the current and upcoming generations of QC hardware.

Despite these advancements, open challenges remain, particularly in refining subproblem selection strategies and including specific constraint structures. In opposition to the top-down approach taken

in this chapter, we delve into a bottom-up subproblem generation method in Chapter 6. We rigorously prove optimality for this procedure and allow for incorporation of efficient classical algorithms that solve certain substructures.

CHAPTER 6

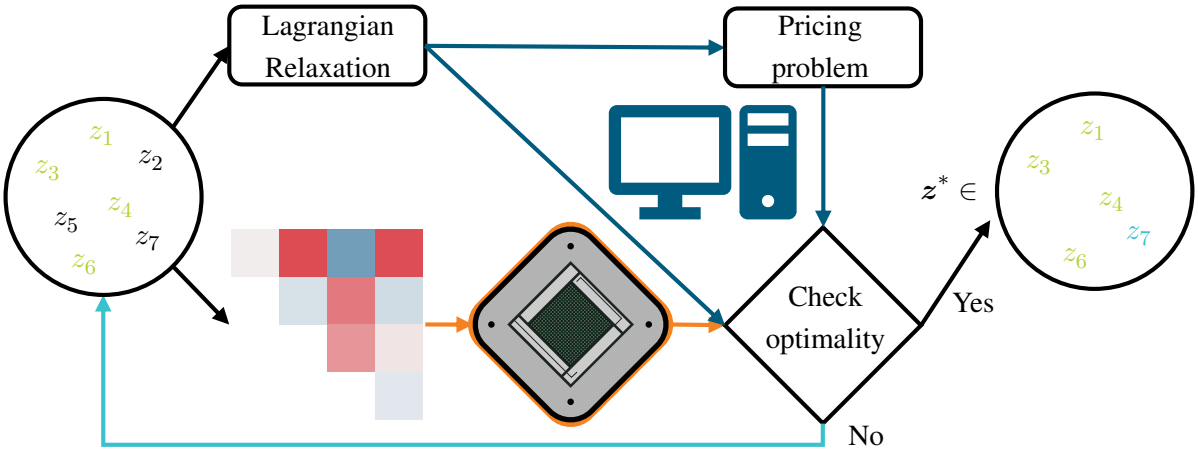
Variable and Constraint Generation

In Chapter 5, we addressed the challenges posed by the limited qubit availability and noise sensitivity of NISQ devices when solving large-scale QUBO problems in QO. Given the high dimensionality of many ML and CO tasks, directly encoding these problems is often infeasible in practice. To mitigate this, we explored problem size reduction techniques with subproblem decomposition and iterative constraint incorporation in form of a recursive D&C algorithm. However, by breaking down the problem into independent subproblems, essential global correlations are sometimes lost, leading to suboptimal solutions when the subsolutions are recombined. While the recursive nature of the method attempts to mitigate this, it cannot capture all dependencies in certain cases. Moreover, it is mainly suited for a certain type of constraints.

In this chapter, a different approach is taken with guarantees to find an optimal solution. Instead of solving the original problem upfront with all possible variables, called Master Problem (MP), it begins with a small subset and dynamically introduces new variables as needed. That is, a Restricted Master Problem (RMP) is iteratively expanded with new decision variables. In classical Linear Programming (LP), this method is known as Column Generation [35], which builds on the hope that many variables take the value zero in an optimal solution and thus do not contribute to the objective function. Specifically, we examine ILP problems, which are introduced in Section 6.2. Variables are added in an iterative fashion and we prove an optimality criterion, which tells us when our current RMP already contains an optimal solution to the original MP. For this criterion, we use bounds on an optimal solution of the given ILP and solve a so-called *pricing* problem, telling us which variable to add next in Section 6.3.1. An overview for the algorithm of iteratively adding variables and constraints is given in Figure 6.1. Further, as we have seen in Section 5.2, incorporating constraints can also lead to an increased problem size. Hence, it is also instructive to iteratively add constraints, as we will examine in Section 6.3.2.

Obtaining efficiently computable bounds on the optimal value of an ILP is problem-specific. For obtaining a lower bound, we use LR, which is efficiently solvable if the correct constraints are integrated into the objective function. QO comes into play in computing an upper bound: binary ILP can be brought into an equivalent QUBO formulation. Using the variable and constraint generation described earlier, we obtain reasonable problem sizes suited for NISQ devices. For efficiently deciding whether our current set of considered variables comprises an optimal solution, the underlying pricing problem has to be efficiently solvable. In practice, this is often the case, e. g., in *shortest path* type problems.

In Section 6.4, we investigate a well-known real-world use-case which exactly fulfills this efficient pricing problem structure. To be specific, we consider MAPF which involves computing collision-free



(a) Init. set. (b) Solve with CPU. (c) Solve with QPU. (d) Add variable. (e) Optimal.

Figure 6.1: Overview of our proposed bottom-up variable generation algorithm for iteratively adding new variables to control the problem size. First, a set of variables is initialized (a), leading to a low-dimensional RMP. Optimizing the LR results in optimal Lagrangian dual variables and a lower bound on the optimal value. Using the dual variables, a pricing problem is solved classically which can be done efficiently for certain problem structures, such as finding shortest paths (b). A variable not contained in the RMP with minimal reduced cost is obtained by solving the pricing problem. In parallel, an upper bound on the optimal value is obtained by solving a QUBO formulation of the RMP with quantum hardware (c). If an optimality criterion dependent on the obtained bounds and the minimal reduced cost is fulfilled, we obtain equivalence between the RMP and the underlying (potentially way higher-dimensional) MP (e), that is an optimal solution $\boldsymbol{z}^*$ is already in our generated variables. If not, the variable with minimal reduced cost is added to the RMP and steps (b)-(c) are repeated, until we obtain equivalence to the MP.

paths for numerous agents simultaneously, posing significant computational challenges, especially in large-scale scenarios. These challenges are increasingly relevant in real-world applications, such as future Unmanned Aerial Vehicle (UAV) traffic in urban areas, where parcel delivery demands are expected to require coordinating tens of thousands of flight paths [213]. Our approach iteratively adds paths and constraints, solving QUBO formulations to determine an upper bound and identify the optimal solution. We present an optimality criterion tailored towards MAPF and introduce hardware-aware QUBO formulations leveraging conflict graphs for parallel solvability.This formulation not only leads to a sparse QUBO matrix, but also avoids the introduction of slack variables for incorporating the constraints. Extensive benchmarking demonstrates the superiority of our method over prior QUBO-based approaches and classical MAPF solvers using NISQ hardware.

6.1 Related Work

Our algorithms are based on the Branch-and-Cut-and-Price (BCP) idea from [214]. Despite their optimality, such algorithms require a sophisticated branching strategy and are not guaranteed to find a good solution in reasonable time. An anytime adaption of MAPF BCP has already been investigated [215], but further investigation of optimal anytime algorithms is of great interest. Heuristics are used for avoiding exponentially many branching steps, leading to efficient suboptimal algorithms [216]. However, such rounding heuristics can lead to unsatisfying results, making the investigation of QC for this task

intriguing.

QC is a promising candidate for large-scale planning problems [57, 217]. In the area of multi-agent problems, flow-problem formulations have been investigated [218–221]. These methods are edge-based, that is each edge in the spatio-temporal graph is represented by a qubit. Even though certain constraints can be conveniently integrated into the quantum state in this framework, the problem size is way beyond current quantum hardware capabilities and also infeasible for near-term devices.

Instead of representing all edges in the given graph, [222] take a different approach considering which path to choose as decision variables. They introduce a QUBO formulation for the *Unsplittable Multi-Commodity Flow* problem by directly integrating the inequality constraints. Similar to BCP, they iteratively add paths to their problem. However, they present no theoretically sound criterion on when to stop, but have to rely on suboptimal heuristics. Furthermore, the large amount of constraints have to be incorporated into the QUBO formulation which either leads to a huge number of auxiliary variables or the need for an iterative optimization scheme to adapt Lagrangian parameters [198]. [164, 223] circumvent this problem by using conflict graphs for representing possible constraints. However, their methods are not directly applicable to the MAPF setting, since only the starting times of pre-planned trajectories are optimized.

We combine the ideas of an iteratively expanding path-based approach with the concept of conflict graphs. A pricing criterion tells us when all variables are included which are part of an optimal solution. The proof is based on [224], which however assumes negativity on reduced costs. We loosen this assumption and adapt it for general applicability to MAPF.

6.2 Integer Linear Programming

Instead of looking at a quadratic objective as given in QBP, we consider linear objectives in this chapter. In particular, we are concerned with *Integer Linear Programming* (ILP) problems, which take the following form.

Definition 6.1 (ILP)**.** Let $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{A} \in \mathbb{Z}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{Z}^m$. ILP is defined as finding the solution to the following optimization problem

$$\min_{\boldsymbol{z} \in \mathbb{Z}^n} \ \boldsymbol{c}^\top \boldsymbol{z} \tag{6.1a}$$

$$\text{s.t. } \boldsymbol{A}\boldsymbol{z} \preceq \boldsymbol{b} \ , \tag{6.1b}$$

Solving Equation (6.1) is NP-hard and thus it is often intractable to find an optimal solution, especially for a large n. A quite similar problem is given by *Linear Programming* (LP).

Definition 6.2 (LP)**.** Let $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{A} \in \mathbb{Z}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{Z}^m$. LP is defined as finding the solution to the following optimization problem

$$\min_{\boldsymbol{z} \in \mathbb{R}^n} \ \boldsymbol{c}^\top \boldsymbol{z} \tag{6.2a}$$

$$\text{s.t. } \boldsymbol{A}\boldsymbol{z} \preceq \boldsymbol{b} \ , \tag{6.2b}$$

Given an ILP, we denote its *relaxation* as the LP using the same costs and constraints. In fact, the optimum of the relaxation is always smaller than the optimum of the original ILP. Since we are faced

with a linear objective in Equation (6.1), the constraints determine the hardness of the problem. For example, the LP relaxation of an ILP has integer solutions if the set $\{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{z} \preceq \boldsymbol{b}\}$ is convex, i. e., $\{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}\} = \mathrm{CH}(\{\boldsymbol{z} \in \mathbb{Z}^n : \boldsymbol{A}\boldsymbol{z} \preceq \boldsymbol{b}\})$, where CH denotes the *convex hull*. This leads to the idea to split the constraints into *easy* and *hard* constraints. A formalization is given by introducing the LR [36].

Definition 6.3 (LR)**.** Let $\boldsymbol{c} \in \mathbb{R}^n$, constraints described by $\boldsymbol{A} \in \mathbb{Z}^{k \times n}$, $\boldsymbol{b} \in \mathbb{Z}^k$, $\boldsymbol{D} \in \mathbb{Z}^{m \times n}$ and $\boldsymbol{d} \in \mathbb{Z}^m$ and consider the ILP of the form

$$\min_{\boldsymbol{z} \in \mathbb{Z}^n} \boldsymbol{c}^\top \boldsymbol{z} \tag{6.3a}$$

$$\text{s.t. } \boldsymbol{A}\boldsymbol{z} \preceq \boldsymbol{b} \tag{6.3b}$$

$$\boldsymbol{D}\boldsymbol{z} \preceq \boldsymbol{d}\,. \tag{6.3c}$$

With $\boldsymbol{\lambda} \in \mathbb{R}^m_+$, the *partial Lagrangian* is given by

$$\mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}) \coloneqq \boldsymbol{c}^\top \boldsymbol{z} + \boldsymbol{\lambda}^\top (\boldsymbol{D}\boldsymbol{z} - \boldsymbol{d})\,.$$

The LR of the ILP problem given in Equation (6.3) is defined as finding the solution to the following optimization problem

$$\mathcal{L}(\boldsymbol{\lambda}) \coloneqq \min_{\boldsymbol{z} \in \mathbb{Z}^n} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}) \tag{6.4a}$$

$$\text{s.t. } \boldsymbol{A}\boldsymbol{z} \preceq \boldsymbol{b}\,. \tag{6.4b}$$

That is, we "absorb" the hard constraints into the objective function.

Correctly splitting all constraints into hard and easy can lead to Equation (6.4) being easily solvable. Let $v(\cdot)$ denote the optimal value of a programming problem mentioned before. It then holds that (c.f. [36])

$$\mathcal{L}(\boldsymbol{\lambda}) \leq v(\mathrm{ILP}),\ \forall \boldsymbol{\lambda} \in \mathbb{R}^m_+\,.$$

Thus, we can easily find a lower on the optimum of the ILP problem, which is hard to compute. To obtain as much information as possible, we want to maximize this lower bound.

Definition 6.4 (LD)**.** Let $\boldsymbol{c} \in \mathbb{R}^n$, constraints described by $\boldsymbol{A} \in \mathbb{Z}^{k \times n}$, $\boldsymbol{b} \in \mathbb{Z}^k$ and $\boldsymbol{D} \in \mathbb{Z}^{m \times n}$, $\boldsymbol{d} \in \mathbb{Z}^m$. The Lagrangian Dual (LD) of the ILP problem given in Equation (6.3) is defined as the maximization of the LR

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}^m_+} \mathcal{L}(\boldsymbol{\lambda})\,. \tag{6.5}$$

The LD can be efficiently computed, e. g., using a *subgradient* method [225]. Interestingly, the following inequalities hold, which combine the optimal values of LP, ILP and LD.

Theorem 6.1 (ILP bounds [36])**.** *Let $\boldsymbol{c} \in \mathbb{R}^n$, constraints described by $\boldsymbol{A} \in \mathbb{Z}^{k \times n}$, $\boldsymbol{b} \in \mathbb{Z}^k$ and $\boldsymbol{D} \in \mathbb{Z}^{m \times n}$, $\boldsymbol{d} \in \mathbb{Z}^m$. Then, the following inequalities hold true for an ILP of the form given in Equation* (6.3)

$$v(LP) \leq \max_{\boldsymbol{\lambda} \in \mathbb{R}^m_+} \mathcal{L}(\boldsymbol{\lambda}) \leq v(ILP)\,. \tag{6.6}$$

Setting $\mathcal{Z} = \{\boldsymbol{z} \in \mathbb{Z}^n : \boldsymbol{A}\boldsymbol{z} \preceq \boldsymbol{b}\}$ *and* $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{D}\boldsymbol{x} \preceq \boldsymbol{d}\}$*, equality in Equation* (6.6) *holds under the following conditions*

$$\mathrm{CH}(\mathcal{Z} \cap \mathcal{X}) = \mathrm{CH}(\mathcal{Z}) \cap \mathcal{X} \;\Leftrightarrow\; v(ILP) = \max_{\boldsymbol{\lambda} \in \mathbb{R}^m_+} \mathcal{L}(\boldsymbol{\lambda}), \tag{6.7a}$$

$$\mathrm{CH}(\mathcal{Z}) = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}\} \;\Rightarrow\; v(LP) = \max_{\boldsymbol{\lambda} \in \mathbb{R}^m_+} \mathcal{L}(\boldsymbol{\lambda})\ . \tag{6.7b}$$

Thus, if Equation (6.7b) holds, we can easily obtain the value of LD with using an efficient algorithm for LP, such as the *simplex algorithm* [226].

6.3 Column Generation for Binary Linear Programs

Bridging the gap to QUBO, we assume that our problem at hand can be formulated as *binary linear programs*, that is

$$\text{MP}: \min_{\boldsymbol{z} \in \mathbb{B}^N} \boldsymbol{c}^\top \boldsymbol{z} \tag{6.8a}$$

$$\text{s.t. } \boldsymbol{A}\boldsymbol{z} \preceq \boldsymbol{b} \tag{6.8b}$$

$$\boldsymbol{D}\boldsymbol{z} \preceq \boldsymbol{d}\ , \tag{6.8c}$$

with $\boldsymbol{A} \in \mathbb{Z}^{k \times N}$, $\boldsymbol{b} \in \mathbb{Z}^k$, $\boldsymbol{D} \in \mathbb{Z}^{M \times N}$ and $\boldsymbol{d} \in \mathbb{Z}^M$ with $M, N, k \in \mathbb{N}$. We call Equation (6.15) the *master problem* (MP) and use the capital letters M and N to stress that these values can be very large. That is, assume we are faced with many constraints and variables. As we have seen in Chapter 5, solving such problems with NISQ devices is intractable. Thus, we consider the *restricted master problem* (RMP)

$$\text{RMP}: \min_{\boldsymbol{z} \in \mathbb{B}^N_{\mathcal{J}^c \leftarrow \mathbf{0}}} \boldsymbol{c}^\top \boldsymbol{z} \tag{6.9a}$$

$$\text{s.t. } \boldsymbol{A}\boldsymbol{z} \preceq \boldsymbol{b} \tag{6.9b}$$

$$\boldsymbol{D}_{\mathcal{I}}\boldsymbol{z} \preceq \boldsymbol{d}_{\mathcal{I}}\ , \tag{6.9c}$$

with $\mathcal{I} \subset [M]$, $|\mathcal{I}| = m$, $\mathcal{J} \subset [N]$, and $|\mathcal{J}| = n$. We only optimize over a subset of variables ($n \ll N$) and constraints ($m \ll M$), drastically reducing the problem size.

To get equivalence between Equation (6.9) and Equation (6.8), we use a two-loop iterative optimization scheme: the outer loop decides which constraints are not fulfilled and should be added to our problem and the inner loop decides which variables should be added not included in the RMP (see Figure 6.1 and Algorithm 6). This procedure builds with the hope that we already find a (nearly) optimal solution with not exploring the whole search space, both in terms of decision variables n and number of constraints m. The mathematical framework describing this technique is called *column/row generation* [35]. The columns are identified with the decision variables and the rows with the constraints. It alternatingly solves a high-level *restricted master problem* (RMP) and low-level *pricing* and *separation* problems (PP and SP), which decide which variables/constraints to add to the MP. If solving the PP/SP tells us to not add any more variables, the solution to the MP is optimal by construction.

This method was developed for LP without the restriction of integrality of the variables—in our

Algorithm 6 QUBOANDPRICEANDCUT

Input: Initial set of decision variables $\mathcal{J} \subset [N]$
Output: Optimal feasible solution

$\mathcal{I} \leftarrow \emptyset$
while $\boldsymbol{z}$ is infeasible **do** ▷ Separation
  Add violated constraints to $\mathcal{I}$
  while not Equation (6.10) **do** ▷ Pricing
    $j^* \leftarrow \arg\min_{j \in \mathcal{J}^c} \bar{c}_j(\boldsymbol{\lambda}, \mathcal{I})$
    $\mathcal{J} \leftarrow \mathcal{J} \cup \{j^*\}$
    $\boldsymbol{\lambda} \leftarrow$ OPTIMIZELAGRANGIAN$(\mathcal{J}, \mathcal{I})$
    $\boldsymbol{z} \leftarrow$ OPTIMIZEMASTER$(\mathcal{J}, \mathcal{I})$ ▷ QUBO
  end while
end while

case we do not optimize over the continuous domain $[0, 1]$ but over the binary values $\{0, 1\}$. To cope with these kind of ILP problems, *branching* methods, such as BCP [227], are used. In the worst case, exponentially many branching steps are needed. We circumvent this problem by considering QUBO solutions to the ILP MP, instead of the relaxed LP version. Ideally, we want that $n \ll N$, while an optimum of RMP is also an optimum of MP. An iterative algorithm for variable and constraint generation can be found in Algorithm 6 and Figure 6.1. How to add new variables to our problem and check whether a solution is equivalent is governed in the next section.

6.3.1 Pricing

For obtaining an optimality criterion, we first define the *reduced cost*.

Definition 6.5 (Reduced cost)**.** Let $\boldsymbol{c} \in \mathbb{R}^n$, constraints described by $\boldsymbol{D} \in \mathbb{Z}^{m \times n}$, $\boldsymbol{d} \in \mathbb{Z}^m$ and $\boldsymbol{\lambda} \in \mathbb{R}^m_+$. The *Lagrangian reduced cost vector* is defined as

$$\bar{\boldsymbol{c}}(\boldsymbol{\lambda}) \coloneqq \boldsymbol{c} + \boldsymbol{\lambda}^\top \boldsymbol{D} \,.$$

In classical column generation, the MP and RMP are equivalent if the minimal reduced cost is ≤ 0. In the more restricted ILP case, we obtain a looser bound for guaranteeing optimality.

Theorem 6.2. *Let $\boldsymbol{c} \in \mathbb{R}^N_+$, $\boldsymbol{A} \in \mathbb{Z}^{k \times N}$, $\boldsymbol{b} \in \mathbb{Z}^k$, $\boldsymbol{D} \in \mathbb{Z}^{M \times N}_+$, $\boldsymbol{d} \in \mathbb{Z}^M$ with $M, N, k \in \mathbb{N}$, $\mathcal{I} = [M]$ and $\mathcal{J} \subset [N]$. Let $\hat{v}$ be the objective value of a feasible solution $\bar{\boldsymbol{z}}$ to* RMP *($\boldsymbol{D}\bar{\boldsymbol{z}} \preceq \boldsymbol{d}$, $\boldsymbol{A}\bar{\boldsymbol{z}} \preceq \boldsymbol{b}$ and $\hat{v} = \boldsymbol{c}^\top \bar{\boldsymbol{z}}$), $\boldsymbol{\lambda} \in \mathbb{R}^m_+$. If v*(RMP) $\neq v$(MP) *then*

$$\min_{j \in \mathcal{J}^c} \bar{c}_j(\boldsymbol{\lambda}) \geq \hat{v} + \boldsymbol{\lambda}^\top \boldsymbol{d} \,. \tag{6.10}$$

Proof. We give a proof by showing that v(RMP) $= v$(MP) holds if Equation (6.10) does not hold. The variables not in RMP are implicitely assumed to take the value 0. Assume now, that one variable not

included in the RMP takes value 1, i.e., $\mathbf{1}^\top \boldsymbol{z}_{\mathcal{J}^c} \geq 1$. It follows that

$$
\begin{aligned}
v(\text{MP}) &= \min_{\boldsymbol{z} \in \mathbb{B}^N} \left\{ \boldsymbol{c}^\top \boldsymbol{z} : \boldsymbol{D}\boldsymbol{z} \preceq \boldsymbol{d},\ \boldsymbol{A}\boldsymbol{z} \preceq \boldsymbol{b},\ \mathbf{1}^\top \boldsymbol{z}_{\mathcal{J}^c} \geq 1 \right\} && \text{(6.11a)} \\
&\geq \min_{\boldsymbol{z} \in \mathbb{B}^N} \left\{ \mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}) : \boldsymbol{A}\boldsymbol{z} \preceq \boldsymbol{b},\ \mathbf{1}^\top \boldsymbol{z}_{\mathcal{J}^c} \geq 1 \right\} && \text{(6.11b)} \\
&= \min_{\boldsymbol{z} \in \mathbb{B}^N} \left\{ \bar{\boldsymbol{c}}(\boldsymbol{\lambda})^\top \boldsymbol{z} - \boldsymbol{\lambda}^\top \boldsymbol{d} : \boldsymbol{A}\boldsymbol{z} \preceq \boldsymbol{b},\ \mathbf{1}^\top \boldsymbol{z}_{\mathcal{J}^c} \geq 1 \right\} && \text{(6.11c)} \\
&\geq \min_{\boldsymbol{z} \in \mathbb{B}^N} \left\{ \bar{\boldsymbol{c}}(\boldsymbol{\lambda})^\top \boldsymbol{z} : \mathbf{1}^\top \boldsymbol{z}_{\mathcal{J}^c} \geq 1 \right\} - \boldsymbol{\lambda}^\top \boldsymbol{d} && \text{(6.11d)} \\
&\geq \min_{j \in \mathcal{J}^c} \bar{c}_j(\boldsymbol{\lambda}) - \boldsymbol{\lambda}^\top \boldsymbol{d} && \text{(6.11e)} \\
&\geq \hat{v} \geq v(\text{RMP}) \ , && \text{(6.11f)}
\end{aligned}
$$

where Equation (6.11f) holds since we assume that Equation (6.10) does not hold. We follow that no feasible solution to MP with $\mathbf{1}^\top \boldsymbol{z}_{\mathcal{J}^c} \geq 1$ can be better than an optimal solution to RMP. Since $v(\text{MP}) \leq v(\text{RMP})$ is always true, we conclude $v(\text{MP}) = v(\text{RMP})$. Hence, if $v(\text{MP}) \neq v(\text{RMP})$ then Equation (6.10) holds. □

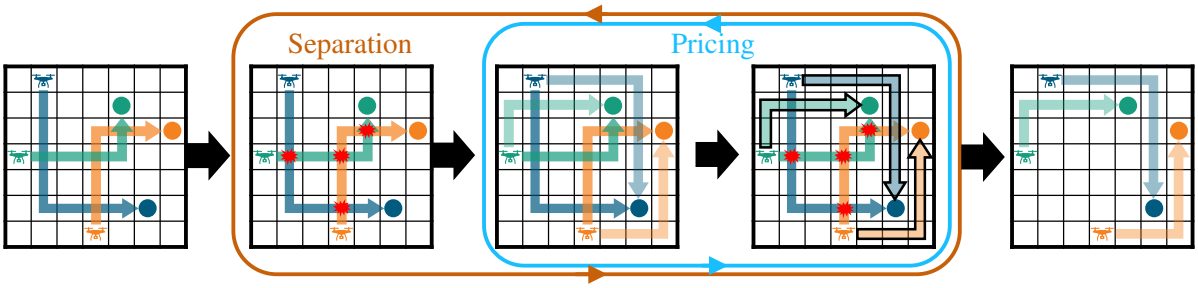
Using the optimality criterion stated in Theorem 6.2, we can decide whether the variables in the RMP contain the ones for an optimal solution to the MP. The given is rather loose, since we only have very few assumptions on the underlying problem. We completely ignore the "easy" constraints $\boldsymbol{A}\boldsymbol{z} \preceq \boldsymbol{b}$ in Equation (6.11d). In practice, we are often faced with a special structure of the constraints, such we can find a tighter bound for the optimality criterion, as we will see in Section 6.4. The tighter the bound, the less variables are added to the RMP for reaching equivalence to the MP. Which variables to add is governed by the pricing problem (PP) which aims to find a variable not included in the RMP with optimal reduced costs

$$
\text{PP} : \min_{j \in \mathcal{J}^c} \bar{c}_j(\boldsymbol{\lambda}) \ . \tag{6.12}
$$

Even though the optimization domain $\mathcal{J}^c$ might be huge, the problem in Equation (6.12) can still be efficiently solvable. For example, finding shortest paths allows efficient enumeration of a number of near-optimal solutions, which will be used in Section 6.4. Instead of relying on optimal Lagrange parameters, our theorem allows any choice of $\boldsymbol{\lambda} \in \mathbb{R}^m_+$. How to obtain these parameters depends on the underlying problem structure. For example, if our constraints are convex, we can obtain optimal parameters by solving the LP relaxation of the LD.

6.3.2 Separation

In addition to adding new decision variables to our RMP, we can take a similar approach with handling the constraints. Starting off with no inequality constraints, $\mathcal{I} = \emptyset$, we can iteratively add them to the RMP, reducing computational overhead. Given a feasible solution $\bar{\boldsymbol{z}}$ to the RMP, we check whether it is also feasible for the MP. That is, we add the row/constraint $\boldsymbol{D}_i \boldsymbol{z} \leq 1$ to the RMP if $\boldsymbol{D}_i \bar{\boldsymbol{z}} > 1,\ i \in [M]$. We investigate the performance of our proposed algorithm in the next section, where we consider the well-known MAPF problem.



(a) Init. paths. (b) Identify conflicts. (c) Generate paths. (d) Solve QUBO. (e) Solution.

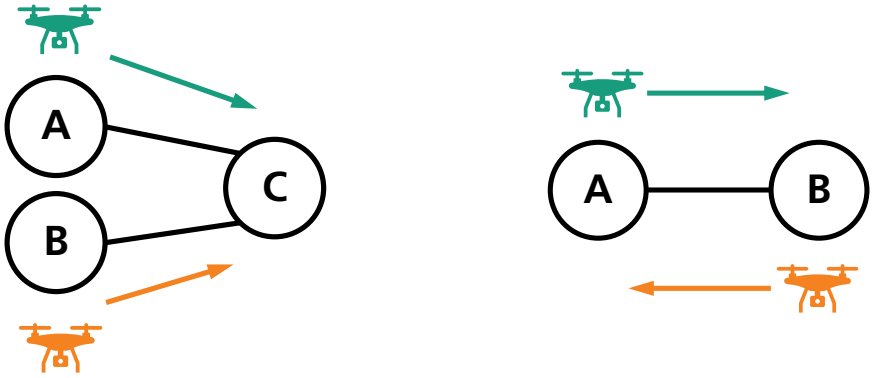
Figure 6.2: Schematic visualization of our quantum MAPF algorithm. (a) First, initial paths are generated for every agent with possible conflicts. (b) We enter the outer loop (separation), where we identify conflicts between paths and add them to the problem. (c) In the pricing step, we generate new paths for every agent and (d) find the best set of paths by solving a QUBO problem. This inner loop is repeated until adding a new path cannot improve the solution quality, while the outer loop is terminated when our set of chosen paths has no conflicts. (e) By construction, a conflict-free optimal set of paths is returned.

6.4 Application: Multi-Agent Pathfinding

Emerging domains of large-scale resource allocation problems, such as assigning road capacity to vehicles, warehouse management or 3D airspace to unmanned aerial vehicles (UAVs), often require multi-agent pathfinding (MAPF) [37, 228, 229] to determine feasible allocations. MAPF involves calculating non-colliding paths for a large number of agents simultaneously, presenting significant computational challenges in realistically sized scenarios.

The scalability of optimal state-of-the-art MAPF solvers is limited, since finding optimal solutions is NP-hard [230]. Thus one often falls back to suboptimal methods, with the most prominent one being *Local Neighborhood Search* (LNS) [231–233]. Even though such methods are computationally efficient in finding a feasible solution, the solution quality can be insufficient which implies the urge for optimal solution methods. By reformulating MAPF as a multi-commodity flow problem, it can be solved optimally via ILP. However, the reduction uses an inefficient representation of the problem setting in terms of space complexity and is only effective on small instances. More popular techniques for optimal MAPF include *Conflict-based Search* [230] and BCP [214]. CBS is a two-level procedure, with splitting a search tree based on detected conflicts between agents and subsequent replanning. This tree is explored with best-first search until a collision-free node is found. BCP takes a different approach of considering a (possibly infeasible) solution which is then refined iteratively by successively adding paths and constraints. Similarly to CBS, BCP is a two-level algorithm: on the low level it solves a series of single-agent pathfinding problems, while the high level uses ILP to assign feasible paths to agents. While low level single agent pathfinding can be performed efficiently, the high level problems of both CBS and BCP remain NP-hard.

In this section, we investigate the use of QC for MAPF by constructing two optimal hybrid quantum-classical algorithms based on QUBO subproblems. To the best of our knowledge, these are the first quantum algorithms for MAPF. We call our algorithms QUBO-and-Price (QP) and QUBO-and-Cut-and-Price (QCP) which are based on the idea of BCP. We iteratively add paths (and constraints) to the problem and solve a QUBO, which leads to the applicability of QC. This gives us an upper bound on the best possible solution and a stopping condition tells us when our set of paths contains the optimal



(a) Vertex conflict.

(b) Swapping edge conflict.

Figure 6.3: The conflicts we are using for our MAPF setup.

solution. An overview of QCP can be found in Figure 6.2.

6.4.1 Different Problem Formulations

The input to the MAPF problem is a set of agents A, a weighted undirected graph $G = (V, E)$ and an origin-destination pair for each agent. The graph G captures the underlying environment, where all possible agent states (e.g. location) are represented by V and E can be regarded as valid transitions from one state to another, with an underlying cost. V already captures environmental constraints, such as possible obstacles, while E captures motion constraints of the agents, e.g., velocity and maximum turning rates of UAVs. The goal of MAPF is now to find optimal paths from the origin to the destination for each agent, s.t. they avoid pairwise conflicts. For rating optimality, we use the objective of the sum of all weighted path lengths. We here consider classical collision conflicts, that is the *vertex conflict* and the *(swapping) edge conflict*. A vertex conflict between two agents exists if they move to the same node at the same time, while an edge conflict prohibits two agents to use the same edge at the same time. Introducing the time component, we allow the agents to start at different points in time, while also giving them the opportunity to wait a certain location.

For finding a solution to the MAPF problem, a spatio-temporal directed graph formulation is typically used. That is, we define $G_T = (V_T, E_T)$ as the graph, with nodes $v = (s, t) \in V \times \{1, \dots, T\}$ and edges $e = (v, v') = ((s, t), (s', t + 1)) \in V_T \times V_T$ with weights $w_e = w_{(s,s')}$, where $(s, s') \in E$ and $T \in \mathbb{N}$ is a maximum allowed time horizon. We define the reverse edge of $e = (s, t), (s', t + 1)$ as $\bar{e} \coloneqq (s', t), (s, t + 1)$ for representing edge conflicts. G_T is an acyclic weighted directed graph with $|V_T| = T\,|V|$ and $|E_T| = \mathcal{O}\,(ST\,|V|)$, where S is the average number of possible state transitions. For example, if we consider the classical two-dimensional grid environment, an agent has five possible state transitions, i.e., wait, go north, go east, go south and go west.

Every agent $a \in A$ is obliged to find a path in G_T from origin $(o_a, t_a) \in V_T$ to destination $(d_a, t_a + T_a) \in V_T$ for a starting time $t_a \in [T]$ and $T_a \leq T - t_a$. A path p is a sequence of edges $p \coloneqq (e_1, \dots, e_{T_a - 1})$, s.t. $e_t = ((x, t_a + t), (y, t_a + t + 1))$, $e_{t+1} = ((y, t_a + t + 1), (z, t_a + t + 2))$, $e_1 = ((o_a, t_a), (s, t_a + 1))$ and $e_{T_a} = ((s, t_a + T_a - 1), (d_a, t_a + T_a))$. The cost/length c_p of a path p is defined as the sum of all its edge weights, i.e., $c_p \coloneqq \sum_{e \in p} w_e$. Specifically, we go over to the

mathematical description of the MAPF objective.

Edge-Based Formulation The MAPF problem with vertex and edge conflicts can be formulated in the ILP paradigm, indicating a used edge e by agent a with a binary variable $z_e^a \in \mathbb{B}$

$$\min_{\boldsymbol{x}\in\mathbb{B}^l} \sum_{a\in A}\sum_{e\in E_T} w_e z_e^a \tag{6.13a}$$

$$\text{s.t.} \sum_{e=(i,j)} z_e^a - \sum_{e'=(j,i)} z_{e'}^a = b_i^a,\ \forall a \in A,\ i \in V_T \tag{6.13b}$$

$$\sum_{a\in A}\sum_{e=(j,i)} z_e^a \leq 1,\ \forall i \in V_T \tag{6.13c}$$

$$\sum_{a\in A} z_e^a + z_{\bar{e}}^a \leq 1,\ \forall e \in E_T\ , \tag{6.13d}$$

where $l = |A||E_T|$ and $b_i^a := \delta_{i,o_a} - \delta_{i,d_a}$. Equation (6.13b) ensures that we have valid paths for every agent, Equation (6.13c) avoids vertex conflicts and Equation (6.13c) prevents edge conflicts.

If we want to reformulate the ILP in Equation (6.13) to a QUBO problem, we have to incorporate the constraints in Equations (6.13b) to (6.13d). This can be done by adding quadratic penalty terms to the objective function Equation (6.13a) which penalizes invalid solutions, i.e., ones that violate at least one of the constraints. Linear equality constraints like Equation (6.13b) can be easily reformulated into such a quadratic penalty (c.f. Proposition 3.1). For general linear inequality constraints we either need to introduce additional slack variables which also have to be optimized or use an iterative method to find optimal Lagrange parameters [198]. More details can be found in Section 6.4.2.

Since we have $|V_T|$ constraints for avoiding vertex conflicts and $|A||E_T|$ constraints for avoiding swapping conflicts, we need to incorporate $|A||E_T|+|V_T|$ constraints for obtaining a QUBO formulation. The number of needed binary variables x_e^a, representing the possible solutions, is $|A||E_T|$. For large environments (S and $|V|$), a large maximum time T or a large number of agents $|A|$, the problem size can increase very quickly, making it infeasible to solve it with current NISQ devices. Due to this limitation, we examine a different encoding.

Path-Based Formulation Instead of representing each edge by a binary variable, we encode each possible path $p \in \mathcal{P}$ for every agent by $z_p \in \mathbb{B}$. An ILP formulation is given by:

$$\min_{\boldsymbol{z}\in\mathbb{B}^N} \sum_{a\in A}\sum_{p\in\mathcal{P}_a} c_p z_p \tag{6.14a}$$

$$\text{s.t.} \sum_{p\in\mathcal{P}_a} z_p = 1,\ \forall a \in A \tag{6.14b}$$

$$\sum_{a\in A}\sum_{p\in\mathcal{P}_a} x_v^p z_p \leq 1,\ \forall v \in V_T \tag{6.14c}$$

$$\sum_{a\in A}\sum_{p\in\mathcal{P}_a} \left(y_e^p + y_{\bar{e}}^p\right) z_p \leq 1,\ \forall e \in E_T\ , \tag{6.14d}$$

where $x_v^p, y_e^p \in \mathbb{B}$ indicate whether path p visits vertex $v \in V_T$ / edge $e \in E_T$, $\mathcal{P}_a$ indicates the set of all possible paths for agent a and $N = |\mathcal{P}|$, $\mathcal{P} = \bigcup_{a \in A} |\mathcal{P}_a|$. Note that x_v^p, y_e^p are constant and we just optimize over z_p. The constraint in Equation (6.14b) ensures that exactly one path is chosen for every agent, while Equations (6.14c) and (6.14d) avoid conflicts, similar to Equations (6.14c) and (6.14d). The problem Equation (6.14) is exactly of the form given in Equation (6.8), i. e., an equivalent more concise formulation is given by

$$\text{MP}: \min_{\boldsymbol{z} \in \mathbb{B}^N} \boldsymbol{c}^\top \boldsymbol{z} \tag{6.15a}$$

$$\text{s.t. } \boldsymbol{A}\boldsymbol{z} = \mathbf{1}_{|A|} \tag{6.15b}$$

$$\boldsymbol{D}\boldsymbol{z} \leq \mathbf{1}_M \ , \tag{6.15c}$$

where $M = |A||E_T| + |V_T|$, $\boldsymbol{c} = (c_p)_{p \in \mathcal{P}}$, $\boldsymbol{D} \in \mathbb{B}^{M \times N}$ and $\boldsymbol{A} \in \mathbb{B}^{|A| \times N}$. Further, let $\mathcal{C}$ denote the set of all constraints, i.e., an element for each possible vertex and edge conflict. Each element $i \in \mathcal{C}$ in this set is identified by a vector

$$\boldsymbol{D}_i = \left(D_{i,p}\right)_{p \in \mathcal{P}}, \ D_{i,p} = \begin{cases} x_i^p, & \text{if } i \in V_T \ , \\ y_i^p + y_{\bar{i}}^p, & \text{if } i \in E_T \ . \end{cases}$$

Then the matrix $\boldsymbol{D}$ captures all inequality constraints from Equation (6.14). Equation (6.15b) ensures that exactly one path is chosen for every agent

$$\boldsymbol{A}_a = (A_{a,p})_{p \in \mathcal{P}}, \ A_{a,p} = \begin{cases} 1, & \text{if } p \in \mathcal{P}_a \ , \\ 0, & \text{if } p \notin \mathcal{P}_a \ . \end{cases}$$

It follows that $\boldsymbol{A}\boldsymbol{z} \preceq \mathbf{1}$ is convex, and hence we identify the corresponding constraints as "easy". We denote Equation (6.14) as the *master problem* (MP).

We have $|V_T|$ constraints for avoiding vertex conflicts and $|E_T|$ constraints for avoiding swapping conflicts. The number of our decision variables representing possible solutions now corresponds to the number of all possible paths for all agents, which is exponential in the maximum time horizon, $N = \mathcal{O}(|A|S^T)$. A large maximum time T or number of agents $|A|$ can make the problem size increase very quickly, making it infeasible to solve it with current NISQ devices.

6.4.2 Adapting Variable Generation to MAPF

Not only the huge number of decision variables poses a challenge for current quantum computers, but also the number of constraints strongly impacts the solvability of the resulting QUBO formulation. We overcome this issue by considering the *restricted master problem* (RMP) which optimizes over a subset of decision variables

$$\text{RMP}: \min_{\boldsymbol{z} \in \mathbb{B}^N_{P \leftarrow \mathbf{0}}} \boldsymbol{c}^\top \boldsymbol{z} \tag{6.16a}$$

$$\text{s.t. } \boldsymbol{A}\boldsymbol{z} = \mathbf{1} \tag{6.16b}$$

$$\boldsymbol{D}\boldsymbol{z} \preceq \mathbf{1} \ , \tag{6.16c}$$

Algorithm 7 QUBOANDPRICEANDCUT

Input: Shortest independent paths P and $\boldsymbol{z} = \mathbf{1}$
Output: Optimal feasible solution

$C \leftarrow \emptyset$
while $\boldsymbol{z}$ is infeasible **do** ▷ Separation
  Add violated constraints to C
  while not Equation (6.18) **do** ▷ Pricing
    $p^* \leftarrow \arg\min_p \bar{c}_p(\boldsymbol{\lambda}, C)$ ▷ Shortest path
    $P \leftarrow P \cup \{p^*\}$
    $\boldsymbol{\lambda} \leftarrow$ OPTIMIZELAGRANGIAN(P, C) ▷ Solve Equation (6.21)
    $\boldsymbol{z} \leftarrow$ OPTIMIZEMASTER(P, C) ▷ QUBO
  end while
end while

with $P_a \subset \mathcal{P}_a$, $n = |P|$, $P = \bigcup_{a \in A} P_a$. To get equivalence between Equation (6.16) and Equation (6.14), we use a two-loop iterative optimization scheme (Algorithm 7), similar to the one given in Algorithm 6. The outer loop decides which constraints are not fulfilled and should be added to our problem and the inner loop optimizes over subsets of all possible paths for every agent, increasing the number of paths in every step (see Figure 6.2 and Algorithm 7). How to add new paths to our problem and an optimality criterion similar to the one Theorem 6.2 but with tighter bounds is given the next section.

Pricing We identify $\boldsymbol{D}\boldsymbol{z} \preceq \boldsymbol{d}$ as the hard constraints and recall the definition of LR

$$\text{LR} : \mathcal{L}_{\text{RMP}}(\boldsymbol{\lambda}) \coloneqq \min_{\boldsymbol{z} \in \mathbb{B}^N_{\mathcal{J}^c \leftarrow \mathbf{0}}} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}) \tag{6.17a}$$

$$\text{s.t. } \boldsymbol{A}\boldsymbol{z} = \mathbf{1}\ , \tag{6.17b}$$

where $L(\boldsymbol{z}, \boldsymbol{\lambda}) \coloneqq \bar{\boldsymbol{c}}(\boldsymbol{\lambda})^\top \boldsymbol{z} - \boldsymbol{\lambda}^\top \mathbf{1}$ and $\bar{\boldsymbol{c}}(\boldsymbol{\lambda}) \coloneqq \boldsymbol{c} + \boldsymbol{\lambda}^\top \boldsymbol{D}$ is the *Lagrangian reduced cost* vector. Note that $\mathcal{L}_{\text{RMP}}(\boldsymbol{\lambda}) \leq v(\text{RMP})$ follows from Theorem 6.1, where $v(\cdot)$ indicates the optimal value of the optimization problem.

Theorem 6.3. *Let $\hat{v}$ be the objective value of a feasible solution $\bar{\boldsymbol{z}}$ to* RMP *($\boldsymbol{D}\bar{\boldsymbol{z}} \preceq \mathbf{1}_M$, $\boldsymbol{A}\bar{\boldsymbol{z}} = \mathbf{1}_{|A|}$ and $\hat{v} = \boldsymbol{c}^\top \bar{\boldsymbol{z}}$), $\boldsymbol{\lambda} \in \mathbb{R}^m_+$. If v*(RMP) $\neq v$(MP) *then*

$$\exists a \in A : \min_{p \in \mathcal{P}_a \setminus P_a} \bar{c}_p(\boldsymbol{\lambda}) - \min_{p \in P_a} \bar{c}_p(\boldsymbol{\lambda}) < \hat{v} - \mathcal{L}_{RMP}(\boldsymbol{\lambda})\ . \tag{6.18}$$

Proof. We give a proof by showing that v(RMP) $= v$(MP) holds if Equation (6.18) does not hold. The path variables not in RMP are implicitely assumed to take the value 0. Assume now, that one variable

not included in the RMP takes value 1 for agent $a \in A$, i.e., $\sum_{p \in \mathcal{P}_a \setminus P_a} z_p \geq 1$. It follows that

$$
\begin{aligned}
v(\text{MP}) &= \min_{\boldsymbol{z} \in \mathcal{Z}} \left\{ \sum_{p \in \mathcal{P}} c_p z_p : \boldsymbol{D}\boldsymbol{z} \preceq \mathbf{1}, \sum_{p \in \bar{P}_a} z_p \geq 1 \right\} && \text{(6.19a)} \\
&\geq \min_{\boldsymbol{z} \in \mathcal{Z}} \left\{ \mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}) : \sum_{p \in \bar{P}_a} z_p \geq 1 \right\} && \text{(6.19b)} \\
&= \min_{\boldsymbol{z} \in \mathcal{Z}} \left\{ \sum_{p \in \mathcal{P}} \bar{c}_p(\boldsymbol{\lambda}) z_p : \sum_{p \in \bar{P}_a} z_p \geq 1 \right\} - \boldsymbol{\lambda}^\top \mathbf{1} && \text{(6.19c)} \\
&= \min_{\boldsymbol{z} \in \mathcal{Z}} \sum_{p \in \mathcal{P} \setminus \mathcal{P}_a} \bar{c}_p(\boldsymbol{\lambda}) z_p + \min_{p \in \bar{P}_a} \bar{c}_p(\boldsymbol{\lambda}) - \boldsymbol{\lambda}^\top \mathbf{1} && \text{(6.19d)} \\
&= \sum_{b \in A \setminus \{a\}} \min_{p \in \bar{P}_b} \bar{c}_p(\boldsymbol{\lambda}) + \min_{p \in \bar{P}_a} \bar{c}_p(\boldsymbol{\lambda}) - \boldsymbol{\lambda}^\top \mathbf{1} && \text{(6.19e)} \\
&= \min_{p \in \bar{P}_a} \bar{c}_p(\boldsymbol{\lambda}) - \min_{p \in P_a} \bar{c}_p(\boldsymbol{\lambda}) + \mathcal{L}(\boldsymbol{\lambda}) && \text{(6.19f)} \\
&\geq \hat{v} \geq v(\text{RMP}) \,, && \text{(6.19g)}
\end{aligned}
$$

with $\bar{P}_a = \mathcal{P}_a \setminus P_a$ and $\mathcal{Z} = \{\boldsymbol{z} \in \mathbb{B}^N \mid \sum_{p \in \mathcal{P}_a} z_p = 1, \ \forall a \in A\}$. Equation (6.19g) holds since we assume that Equation (6.18) does not hold. We follow that no feasible solution to MP with $\sum_{p \in \mathcal{P}_a \setminus P_a} z_p \geq 1$ can be better than an optimal solution to RMP for each $a \in A$. As all paths not included in the RMP are in the set $\bigcup_{a \in A} \mathcal{P}_a \setminus P_a$ and $v(\text{MP}) \leq v(\text{RMP})$ is always true, we conclude $v(\text{MP}) = v(\text{RMP})$. Hence, if $v(\text{MP}) \neq v(\text{RMP})$ then Equation (6.18) holds. □

The bound in Equation (6.18) is tighter than the one in Equation (6.10), since we exploit the specific problem structure. Instead of solving a pricing problem (PP) globally, we resort to independent problems for every agent

$$
\text{PP} : \min_{p \in \mathcal{P}_a \setminus P_a} \bar{c}_p(\boldsymbol{\lambda}), \quad \forall a \in A \,. \tag{6.20}
$$

Even though $\mathcal{P}_a \setminus P_a$ can be exponentially large, Equation (6.20) can be solved efficiently. It boils down to a k *shortest path problem* ($k \leq |P_a|$) on the graph G_T with updated edge weights—λ_e is added to the cost of w_e and $w_{\bar{e}}$ for conflicted edges and λ_v is added to all incoming edges to conflicted vertices in the current solution $\bar{\boldsymbol{z}}$. This can be done efficiently, e.g. using *Yen's algorithm* [234], which dynamically updates the new shortest path using the already computed paths, obtained with A*.

Interestingly, Equation (6.18) generalizes the stopping criterion in classical column generation. The RHS is an upper bound on the optimality gap between the RMP and LD, i.e., $v(\text{RMP}) - v(\text{LD}) \leq \hat{v} - \mathcal{L}(\boldsymbol{\lambda})$. If that gap is 0, we exactly recover the criterion given in [214] given by $\bar{c}_p - \alpha_a < 0$, where $\alpha_a = \min_{p \in P_a} \bar{c}_p(\boldsymbol{\lambda})$ is the dual variable corresponding to the convexity constraint of agent a Equation (6.14b). In typical column generation, the pricing is solved with an optimal dual solution of

the LP relaxation of the RMP [214]. In our case, we solve the *Lagrangian dual* (LD)

$$\text{LD}: \max_{\boldsymbol{\lambda} \in \mathbb{R}_+^m} \mathcal{L}(\boldsymbol{\lambda}) \,. \tag{6.21}$$

Due to standard ILP theory (c.f. [235]), the optimum of LD and the LP relaxation of RMP coincides, since the one-hot constraint is convex. Also, the dual parameters of this LP relaxation exactly correspond to the optimal $\boldsymbol{\lambda}^*$. Instead of relying on optimal Lagrange parameters, our theorem allows any choice of $\boldsymbol{\lambda} \in \mathbb{R}_+^m$.

Separation Similar to Section 6.3.2, we also iteratively add constraints (Equations (6.14c) and (6.14d)) in addition to adding new paths to our RMP. We add the constraint $\boldsymbol{D}_i \boldsymbol{z} \leq 1$ to the RMP if $\boldsymbol{D}_i \bar{\boldsymbol{z}} > 1,\ i \in V_T \cup E_T$, where $\bar{\boldsymbol{z}}$ is a feasible solution to the RMP. However, to ensure that our algorithm is optimal, we need to find an optimal solution w.r.t. the current constraints. If this cannot be guaranteed, Algorithm 7 can diverge to a suboptimal solution. The performance of this procedure is evaluated in Section 6.5.

Solving RMP with QUBO It remains to clarify how to obtain a solution of Equation (6.16). Since we want the solutions to be solvable with QC, we reformulate the constrained problems into QUBO formulations. We examine three different approaches, which all rely on using penalty factors to integrate constraints.

Proposition 6.1 (One-hot QUBO)**.** *Let $\boldsymbol{A} \in \mathbb{B}^{|A| \times n}$ describe the one-hot constraints given in Equation (6.14b). We can find an equivalent QUBO formulation of the given ILP*

$$\min_{\boldsymbol{z} \in \mathbb{B}^n} \boldsymbol{c}^\top \boldsymbol{z} \quad \Leftrightarrow \quad \min_{\boldsymbol{z} \in \mathbb{B}^n} \boldsymbol{c}^\top \boldsymbol{z} + \sum_{a \in A} \omega_a^o \left(\boldsymbol{A}_a^\top \boldsymbol{z} - 1 \right)^2 .$$
$$\text{s.t. } \boldsymbol{A}\boldsymbol{z} = \boldsymbol{1}$$

Setting $\omega_o^a > \max_{p \in P_a} c_p$ always guarantees the equivalence.

Proof. Follows from Proposition 3.1. □

It remains to incorporate the inequality constraints.

Incorporation of Inequality Constraints The first approach inserts slack variables for every inequality constraint, similar to [221].

Proposition 6.2 (Slack QUBO)**.** *Let $\boldsymbol{D} \in \mathbb{B}^{m \times n}$ describe the inequality constraints given in Equations (6.14c) and (6.14d). We can find an equivalent QUBO formulation of the given ILP*

$$\min_{\boldsymbol{z} \in \mathbb{B}^n} \boldsymbol{c}^\top \boldsymbol{z} \quad \Leftrightarrow \quad \min_{\boldsymbol{z} \in \mathbb{B}^n} \boldsymbol{c}^\top \boldsymbol{z} + \sum_{i \in [m]} \omega_i^s \left(\boldsymbol{D}_i^\top \boldsymbol{z} - s_i \right)^2 ,$$
$$\text{s.t. } \boldsymbol{D}\boldsymbol{z} \preceq \boldsymbol{1}$$

with large enough $\omega_i^s > 0$.

Proof. The linear inequality constraint can be reformulated with using a vector of slack variables $\boldsymbol{s} \in \mathbb{B}^m$ with binary entries,

$$\boldsymbol{D}\boldsymbol{z} \preceq \mathbf{1} \Leftrightarrow \boldsymbol{D}\boldsymbol{z} = \boldsymbol{s} \, .$$

The equivalence follows by using Proposition 5.1. □

Using this formulation, however, comes with the overhead of introducing m additional binary variables. One variable is needed for every inequality constraint, leading to a total QUBO dimension of $n + m$. As we are still in the era of noisy intermediate-scale quantum (NISQ) computers, it is better to resort to a QUBO formulation that uses fewer qubits. However, we analyze the performance of real quantum hardware on this QUBO formulation in Section 6.5.

Since $\boldsymbol{D} \in \mathbb{B}^{m \times n}$, we can avoid using slack variables.

Proposition 6.3 (One-half QUBO)**.** *Let $\boldsymbol{D} \in \mathbb{B}^{m \times n}$ describe the inequality constraints given in Equations* (6.14c) *and* (6.14d)*. We can find an equivalent QUBO formulation of the given ILP*

$$\begin{aligned} &\min_{\boldsymbol{z} \in \mathbb{B}^n} \boldsymbol{c}^\top \boldsymbol{z} \quad \Leftrightarrow \quad \min_{\boldsymbol{z} \in \mathbb{B}^n} \boldsymbol{c}^\top \boldsymbol{z} + \sum_{i \in [m]} \omega_i^s \left(\boldsymbol{D}_i^\top \boldsymbol{z} - \frac{1}{2} \right)^2 , \\ &\quad \textit{s.t. } \boldsymbol{D}\boldsymbol{z} \preceq \mathbf{1} \end{aligned}$$

with large enough $\omega_i^s > 0$.

Proof. We can use the following equivalence

$$\boldsymbol{D}\boldsymbol{z} \preceq \mathbf{1} \Leftrightarrow \left| \boldsymbol{D}_i^\top \boldsymbol{z} - \frac{1}{2} \right| = \frac{1}{2}, \ \forall i \in [m] \Leftrightarrow \min_{\boldsymbol{z} \in \mathbb{B}^n} \left\| \boldsymbol{D}\boldsymbol{z} - \frac{1}{2}\mathbf{1} \right\| \, .$$

Similar to Proposition 3.1, we can deduce the result by adding the corresponding penalty term. □

Thus, we do not optimize over any slack variables and reduce the corresponding QUBO size.

Conflict Graph As a third concept, we examine *conflict graphs* (CG).

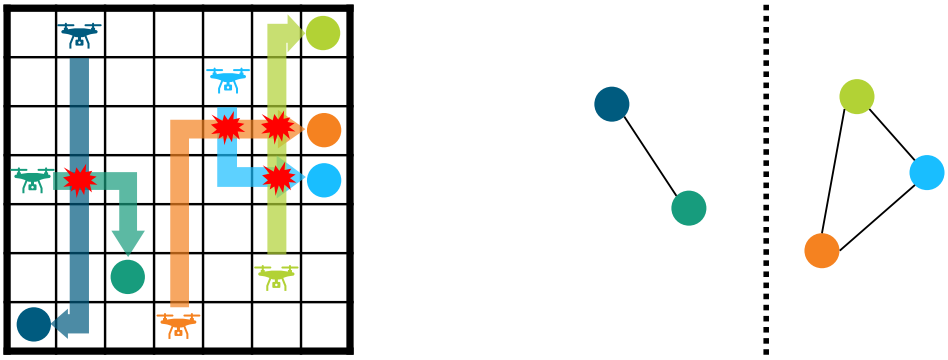
Definition 6.6 (Conflict graph)**.** The vertices of a conflict graph correspond to all paths considered in the problem and the edges indicate whether there is at least one conflict between a pair of paths. That is, its adjacency matrix $\boldsymbol{C} \in \mathbb{B}^{n \times n}$ is given by

$$C_{p,p'} = \begin{cases} 0 & \text{if } x_i^p + x_i^{p'} \leq 1 \text{ and } y_i^p + y_i^p + y_i^{p'} + y_i^{p'} \leq 1 \ \forall i \in V_T \cup E_T, \\ 1 & \text{else.} \end{cases}$$

An exemplary schematic visualization of a conflict graph is given in Figure 6.4.

Proposition 6.4 (Conflict QUBO)**.** *Let $\boldsymbol{D} \in \mathbb{B}^{m \times n}$ describe the inequality constraints given in Equations* (6.14c) *and* (6.14d) *and let $\boldsymbol{C}$ denote the adjacency matrix of the corresponding CG. We can find an equivalent QUBO formulation of the given ILP*

$$\begin{aligned} &\min_{\boldsymbol{z} \in \mathbb{B}^n} \boldsymbol{c}^\top \boldsymbol{z} \quad \Leftrightarrow \quad \min_{\boldsymbol{z} \in \mathbb{B}^n} \boldsymbol{c}^\top \boldsymbol{z} + \omega^c \boldsymbol{z}^\top \boldsymbol{C} \boldsymbol{z} \, , \\ &\quad \textit{s.t. } \boldsymbol{D}\boldsymbol{z} \preceq \mathbf{1} \end{aligned}$$



(a) Conflicted paths of exemplary MAPF problem.

(b) Corresponding disconnected conflict graph.

Figure 6.4: Schematic visualization of a conflict graph, which has two connected components. This leads to the decomposition into independent subproblems with reduced problem size.

with large enough $\omega^c > 0$.

Proof. Assuming that $\boldsymbol{D}\boldsymbol{z} \preceq \mathbf{1}$, we follow

$$\sum_{a \in A} \sum_{p \in \mathcal{P}_a} x_v^p z_p \leq 1,\ \forall v \in V_T \Leftrightarrow x_v^p + x_v^{p'} \leq 1,\ \forall p, p' \in \{p \in P : z_p = 1\},$$

$$\sum_{a \in A} \sum_{p \in \mathcal{P}_a} \left(y_e^p + y_{\bar{e}}^p\right) z_p \leq 1,\ \forall e \in E_T \Leftrightarrow y_e^p + y_{\bar{e}}^p + y_e^{p'} + y_{\bar{e}}^{p'} \leq 1,\ \forall p, p' \in \{p \in P : z_p = 1\}\ ,$$

and thus no penalty occurs for $z_p z_{p'}$, since $C_{p,p'}$. If there exists a row i s.t. $\boldsymbol{D}_i^\top \boldsymbol{z} > 1$, then

$$\sum_{a \in A} \sum_{p \in \mathcal{P}_a} x_v^p z_p > 1,\ \exists v \in V_T \Leftrightarrow x_v^p + x_v^{p'} > 1,\ \exists p, p' \in \{p \in P : z_p = 1\},$$

$$\sum_{a \in A} \sum_{p \in \mathcal{P}_a} \left(y_e^p + y_{\bar{e}}^p\right) z_p > 1,\ \exists e \in E_T \Leftrightarrow y_e^p + y_{\bar{e}}^p + y_e^{p'} + y_{\bar{e}}^{p'} > 1,\ \exists p, p' \in \{p \in P : z_p = 1\}\ .$$

Choosing $\omega^c > \boldsymbol{c}^\top \mathbf{1}$ leads to equivalence. □

Note that the adjacency matrix is square and the dimension is only dependent on the number of considered paths and not on the number of constraints. Since the structure of the CG depends on the underlying problem, it may contain several connected components. Thus, the graph of the QUBO matrix can have multiple connected components, dependent on $\boldsymbol{C}$. These connected components can be seen as smaller instances, giving us the ability to solve them independently. This can lead to large reduction of the considered problem sizes, which is very vital for current QCs, due to the limited number of available qubits. Furthermore, a lower density and well-behaved problem structure can have a huge effect on current quantum hardware, as we will see in Section 6.5.

This leads to two different hybrid quantum-classical MAPF algorithms. QUBO-and-Price (QP) describes generating new paths and stopping when Equation (6.18) is violated (inner loop in Figure 6.2). Even though the QUBO solver might not be optimal, this procedure leads to generating all sufficient paths for obtaining an optimal solution. We denote the extension of iteratively adding constraints to our

problem as QUBO-and-Cut-and-Price (QCP). It is only guaranteed to generate an optimal path set if the QUBO is solved to optimality in every step. However, we also investigate its suboptimal performance in the next section.

6.5 Experimental Evaluation

We compare our algorithms QP and QCP with two state-of-the-art MAPF solvers: BCP [214] and LNS2 [232]. BCP is an optimal algorithm but has also been adapted to anytime, while the suboptimal LNS2 can quickly generate a good solution. The code was taken from their respective public repositories[1][2]. We kept all standard parameters and set a maximum time limit of $180\,s$ for both solvers. The experiments were conducted on a single core of the type Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz.

For QP and QCP we generate the initial paths with *prioritized path planning* (PPP) [236]. That is, every agent is planned successively, with removing edges and nodes of previously planned paths. No clever priorization heuristic is followed, we randomly sample which agent to be chosen next. We use a maximum limit of 30 pricing steps in total and compute optimal Lagrangian parameters $\boldsymbol{\lambda}^*$ with LP in every iteration. We compare the optimal solution of the RMP obtained by an ILP Brach-and-Bound method with the solutions obtained by two different QUBO solvers. As a classical baseline, we use the *Simulated Annealing* (SA) implementation of D-Wave and for real quantum hardware, we run experiments on a D-Wave Advantage_system5.4 quantum annealer (QA). Since both solvers are probabilistic, we generate $1,000$ samples each and use default parameters. We compare the three different QUBO formulations from Section 6.4.2 and denote them by SLACK Proposition 6.2, HALF Proposition 6.3 and CONFLICT Proposition 6.4.

As maps and instances, we use the in the MAPF community well-known MovingAI benchmark [37]. This benchmark includes 33 maps and 25 random scenarios, some of which were utilized in our experiments. Each scenario on each map (with some exceptions) consists of $1,000$ start-goal position pairs. To evaluate a solver on a given scenario, we run it on the first 20, 40, 60, 80 and 100 start-goal pairs for all 25 scenarios and indicate the average performance.

6.5.1 Benchmark Performance

In Figure 6.5, we depict the performance of our two algorithms QP and QCP for four different maps, namely *random-32-32-10*, *maze-32-32-4*, *room-64-64-8* and *den312d*. All performances are shown relative to the best and worst performing configuration, i.e., $v_{\text{rel}} := (v - v_{\text{best}})/(v_{\text{worst}} - v_{\text{best}})$ and averaged over all 25 scenarios. This maps all obtained values to the range $[0, 1]$, where 0 indicates the best and 1 the worst performance, respectively. For QC-QUBO and QCP-QUBO we use the CONFLICT formulation and the SA solver.

The top plot row shows the mean relative upper bound $\hat{v}_{\text{rel}} - v_{\text{rel}}(\text{LD})$ on the optimality gap for a different number of agents. It is not only a measure of solution quality, but it also quantifies the problem size. The higher this gap is, the more new paths are added to our problem during pricing, since it corresponds to the RHS in our optimality criterion Equation (6.18). We can see that optimally solving the RMP (QCP-ILP and QP-ILP) often has a smaller gap than generating a possibly suboptimal solution by a QUBO solver. However, the QUBO methods largely improve upon the base PPP method. Even

[1] https://github.com/Jiaoyang-Li/MAPF-LNS2 (last accessed September 19, 2025)
[2] https://github.com/ed-lam/bcp-mapf (last accessed September 19, 2025)

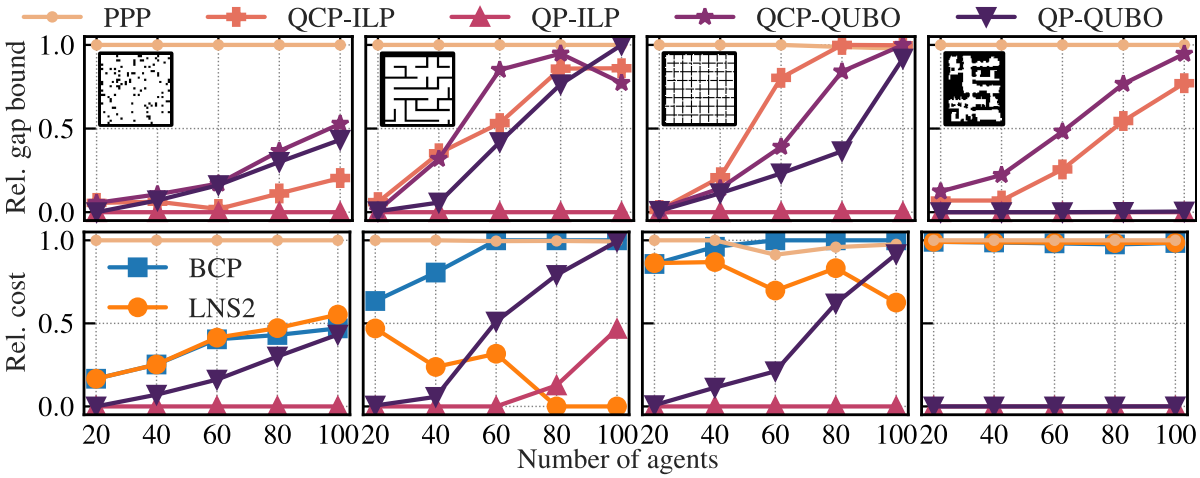


Figure 6.5: Relative performance comparison of our methods QP-ILP, QP-QUBO, QCP-ILP and QCP-QUBO to the baselines PPP, BCP and LNS2 on four different maps with a varying number of agents. The relative upper bound of the optimality gap (top) is shown along with the relative total path costs (bottom) averaged over all 25 scenarios. The lower the better, i.e., a value of 0 corresponds to the best performance, while 1 corresponds to the worst performing algorithm.

though QP clearly outperforms QCP for optimal solving (ILP) of RMP, QCP takes way less constraints into account and is thus computationally more efficient. This is also beneficial for the QUBO solvers, since they can easier generate good solutions for a more well-behaved problem (less constraints).

The mean relative total path cost for the single MAPF instances are depicted in the bottom plot row. We compare the baselines PPP, BCP and LNS2 with QP solving the RMP optimally (QP-ILP) and with QUBO (QP-QUBO). If BCP does not return any solution in the given time window, we set its performance to the worst other performing method (PPP), allowing for a naive anytime comparison. It is evident that QP-ILP is nearly almost optimal, while LNS2 is able to outperform it for 80 and 100 agents for *maze-32-32-4*. BCP is always outperformed by our methods and LNS2 due to its bad anytime performance. However, it is interesting that for no map it is able to find all optimal solutions in the given time frame. QP-QUBO is able to outperform LNS2 in many cases, making our method already applicable without the need of exactly solving the RMP.

The most time consuming step in our algorithm is to solve the QUBO problem. However, a wall-clock time comparison is difficult for quantum devices nowadays, since there is a large communication overhead when using real quantum hardware over cloud services. Nevertheless, assuming perfect communication with the QA, we can generate solutions to a QUBO with an annealing time of around $50\,\mu s$. Due to its probabilistic nature, we have to generate only few thousand samples. In the near future, this could also lead to very good wall-clock time performance.

6.5.2 QUBO Comparison

The effect of using different QUBO formulations for solving the RMP is depicted in Figure 6.6. We consider four different maps (*maze-32-32-4*, *empty-32-32*, *random-32-32-10*, *room-32-32-4*) and compare the results of QP (Section 6.5.2) and QCP (Section 6.5.2) for 20 agents. This leads to varying QUBO sizes between 20 and 400, depending on the underlying map and scenario. We use SA and QA

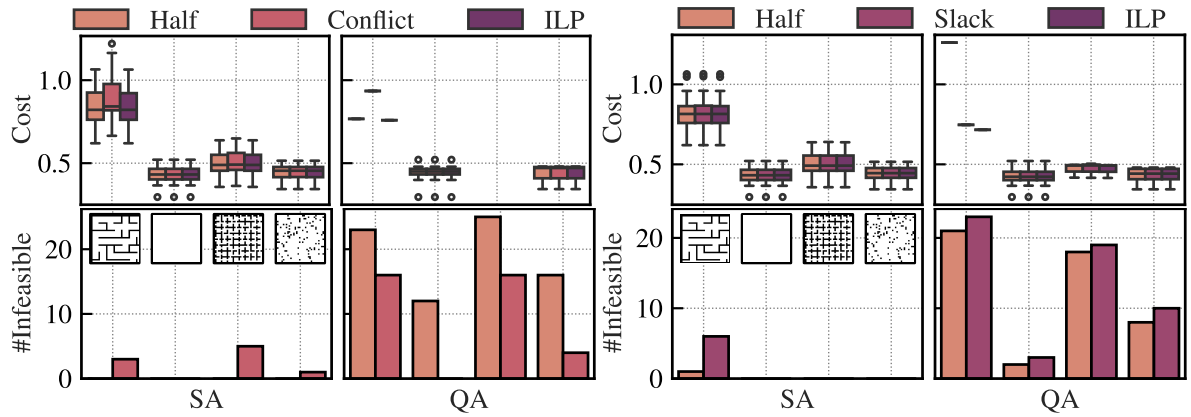


Figure 6.6: Performance comparison of different QUBO formulations for four different maps with 20-agent problems along with the optimal solution, where we run QP (Section 6.5.2) and QCP (Section 6.5.2) for 30 pricing steps. We compare SA and QA by indicating the total path cost of the best sample (top) and the number of infeasible solutions (bottom), i.e. Equations (6.14c) and (6.14d) are violated. The cost is scaled by 10^{-3}.

for solving HALF and CONFLICT for QP and HALF and SLACK for QCP.

In the top row, we show the cost of the best sample obtained by SA and QA over all 25 scenarios, along with the optimal solution (ILP). Only feasible samples are indicated here, that is, only those that adhere to the constraints. For QP, we can see that finding a feasible solution with the HALF QUBO is nearly almost optimal while the solution quality of CONFLICT slightly deteriorates. Comparing HALF and SLACK for QCP, we observe that both solvers are able to find optimal solutions. However, QA has problems finding feasible solutions for the first map for all QUBO formulations.

The number of infeasible solutions returned by SA and QA is depicted in the bottom row. While SA always finds feasible solutions for HALF, it is easier for QA to obtain feasibility with CONFLICT. This is due to the sparsity advantage of CONFLICT over HALF and the corresponding decomposition into independent subproblems. Since the hardware topology of current quantum computers is strongly limited, such properties have a large effect on the solution quality obtained. For QCP, we find that QA finds slightly more feasible solutions with HALF than SLACK. However, we note that only a few separation steps have happened and thus only a small number of constraints have been generated. Using more pricing steps or scaling up the problem size would lead to way more included constraints, making the SLACK QUBO infeasible to solve.

6.6 Conclusion

To address the limitations of NISQ devices in solving large-scale QUBO problems, we explore a variable and constraint generation approach that ensures an optimal solution while maintaining a manageable problem size. Instead of encoding all possible variables upfront in a MP, we iteratively expand a RMP by selectively adding new variables. This approach, inspired by Column Generation in LP, assumes that many variables take a value of zero in the optimal solution, allowing us to introduce only the most relevant ones. Specifically, we prove an optimality criterion for ILP problems based on upper and lower bounds obtained through quantum and classical optimization methods. A classical pricing problem

determines which variable to introduce next, while constraints are incorporated iteratively to avoid excessive problem size growth. We leverage QUBO formulations to obtain upper bounds and LR for lower bounds, efficiently managing problem complexity. This makes our approach well-suited for NISQ devices.

As a real-world application, we applied this approach to MAPF, where the challenge lies in computing collision-free paths for multiple agents simultaneously. Given the high dimensionality of MAPF, our method iteratively adds paths and constraints while solving QUBO formulations to refine upper bounds and ensure optimality. We introduced an optimality criterion tailored towards MAPF and develop hardware-aware QUBO formulations that leverage conflict graphs, enabling parallel solving and reducing computational overhead. This formulation results in a sparse QUBO matrix and eliminates the need for slack variables, further improving efficiency. Extensive benchmarking demonstrates the superiority of our approach over existing QUBO-based methods and classical MAPF solvers when implemented on NISQ hardware.

For our experiments, we initialized our problem with a feasible set of solutions, that is we use prioritized path planning for MAPF. In classical column generation, one often falls back to different methods for generating solutions. Prominent examples are the *big-M* method or *Farkas pricing* [35]. For future work, it would be interesting to investigate the adaption of these methods for the ILP setting. Furthermore, expanding benchmarking to real-world datasets such as UAV traffic or warehouse logistics would be intriguing.

Even though we obtain equivalence between the original MP and the RMP by generating new variables, we have no guarantees on how large the problem dimension of the RMP will be in the end.That is, our problem size can get intractable—in the worst case as large as the MP itself. To circumvent this, we scrutinize an iterative size-adaptable procedure in Chapter 7. Instead of considering subproblems, we improve upon an existing solution by solving a QUBO formulation which naturally incorporates constraints and whose size can be conveniently chosen to match the available NISQ hardware size.

CHAPTER 7

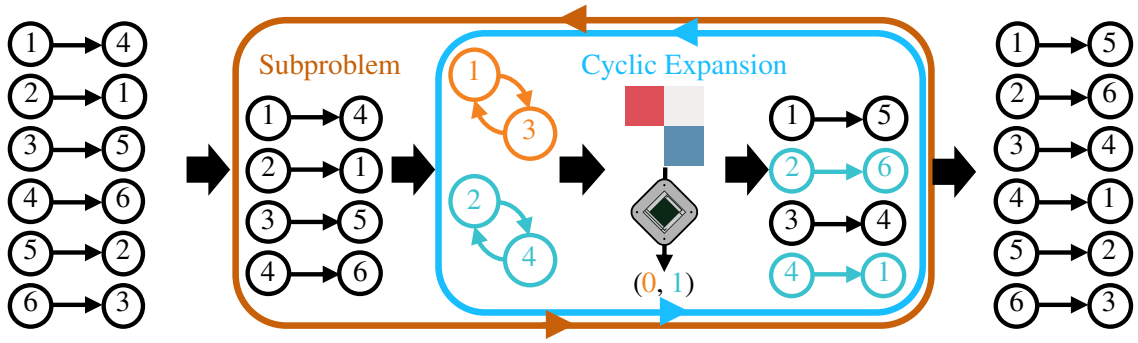
QUBO Size Reduction by Reformulation

In Chapter 6, we introduced a variable and constraint generation approach to address the limitations of NISQ devices in solving large-scale QUBO problems. Rather than considering all decision variables at once, we iteratively expanded a restricted problem by selectively adding relevant variables until optimality is achieved. However, there is no guarantee on the number of variables required to reach equivalence with the original problem, meaning that, in the worst case, the problem size may remain nearly unchanged. Furthermore, it might be the case that many constraints have to be incorporated into the subproblem QUBO formulation. This can lead to a very dense structure of the corresponding QUBO matrix, posing difficulties for NISQ devices with a certain underlying qubit topology.

In this chapter, we overcome these limitations by considering different QUBO formulations of the problem instead of solving subproblems. Reformulating a QUBO problem can significantly reduce its size, making it more suitable for NISQ devices. This can be achieved by leveraging efficient encoding schemes, which can also lead to increased sparsity in the QUBO matrix. In particular, we consider CO problems which aim to find an optimal assignment/permutation. The notoriously hard QAP [237, 238] is discussed in Section 7.2, for which we propose a logarithmic-sized encoding of the binary variables, instead of the well-known quadratic encoding for representing permutation matrices.

Going a step further, we propose an iterative algorithm for solving the unbalanced QAP in Section 7.3, which is inspired by the classical α-expansion algorithm [39], originally developed for computing graph-cuts. The idea is that instead of optimizing over all permutations at once, an iterative optimization over cyclic permutations is carried out which converges towards the original optimization. With this approach, we circumvent the incorporation of constraints for ensuring permutation matrices and can natively integrate further design constraints into the selection procedure of the cycles. The binary variables in the solution of a QUBO decide whether a certain cyclic permutation is applied to our current solution. This encoding allows for arbitrary problem sizes, leading to a scalable and NISQ-aware approach. An overview of our Cyclic Expansion algorithm can be found in Figure 7.1.

We apply our proposed algorithm to the FPGA-placement problem in Section 7.4. It is one of the most time-consuming steps in the implementation pipeline for FPGAs, as well as for the field of chip design in general. Placement is crucial for FPGA design because it directly impacts the performance, power consumption, and resource utilization of the circuit. Due to increased chip size—chip grids consisting of millions of transistors—the need for efficient algorithms is huge. Since QAP is strongly NP-hard, classical algorithms often fail in finding a good solution in reasonable time, while QO is very promising in overcoming these limitations. We apply our proposed cyclic expansion algorithm for



(a) Init. solution. (b) Choose indices. (c) Solve QUBO. (d) Update. (e) Valid solution.

Figure 7.1: Overview of our proposed variable Cyclic Expansion algorithm. Given an initial permutation (a), we choose a subproblem with a suitable size for quantum hardware (b). Random disjoint cyclic permutations are sampled for formulating a QUBO problem, which can then be conveniently solved with a NISQ device (c). The obtained solution indicates which cycles should be applied to our current permutation and it is updated accordingly (d). Steps (c) and (d) are repeated until every possible 2-cycle appeared, while subproblems in (b) are chosen until a certain convergence criterion is met. We end up with a valid solution with better quality than the initialization (e).

iteratively refining a given FPGA-placement. The heterogenous nature of mapping functional blocks to the underlying chip grid locations in FPGA design, can be handily integrated into the choice of the cyclic permutations. Moreover, we can conveniently adapt the problem size to available NISQ-hardware capabilities. Experiments on digital annealing devices and quantum annealers prove the viability of our method.

7.1 Related Work

The QAP is a traditional combinatorial optimization problem [239], [38]. It is NP-hard and no classical algorithms are known which can approximate a solution with quality guarantees in polynomial time. The current state-of-the-art methods for solving the floor planning problem/QAP in FPGA-placement can mainly be divided into three groups: simulated annealing (SA) [240, 241], analytical [40, 242] and partitioning-based [243, 244] approaches. The SA approach can achieve high quality results, especially in terms of subsequent routing time. However, its running time becomes a major drawback when placing a large circuit. Contrary to this, partition-based approaches have a very short running time by recursively partitioning a design. Nevertheless, this might result in bad quality because the problem is solved locally after partitioning. The analytic approach compromises this quality-speed trade-off, by being very fast and showing similar performances to the SA approach. Not every functional block contained in the given net list can be placed anywhere on the chip grid[1], e.g., a LUT cell cannot be placed on an IO location. The analytic methods need a post-processing for incorporating these constraints. There also exist other approaches, e.g., ones who are based on machine learning [245, 246]. All of the aforementioned methods heavily rely on approximations and often need good initializations.

[1] We stick to the term “grid” although the placement problem can indeed be lifted to higher dimensions, e.g., 3-dimensional chips.

The idea of addressing hard combinatorial optimization problem, such as QAP, with quantum computing naturally arises, since quantum computers look promising for overcoming classical limitations. The most advanced research in this field is given in [145, 197, 247], where the paradigm of quantum annealing is applied for solving QAP. Still, the only work we came across in our literature research which is concerned with solving the FPGA-placement problem with quantum computing is [248]. This paper uses a combination of a quantum genetic algorithm and SA and is of rather theoretical nature, without any restrictions on th problem size.

Note that FPGAs are frequently used as control devices in QC hardware [249–251]. However, we apply QC implementations of FPGA designs, which can then be applied to the development of FPGA control units for quantum computers.

7.2 Quadratic Assignment Problem

In particular we examine QAP [237], which aims to find an optimal assignment, also called permutation.

Definition 7.1 (Permutation)**.** Let $X = \{x_1, \dots, x_n\} \subset \mathbb{R}$ and $n \in \mathbb{N}$. A map $\pi : [n] \to [n]$ is a permutation if it is bijective. Permuting an element x_i means, that we apply π to its index i, i.e., $x_i \mapsto x_{\pi(i)}$. Equivalently, it can be depicted by a binary matrix $\boldsymbol{P}$, the so called permutation matrix

$$\boldsymbol{P} \in \mathbb{P}_n \coloneqq \left\{ \boldsymbol{Z} \in \mathbb{B}^{n \times n} : \boldsymbol{Z}\mathbf{1} = \mathbf{1},\ \boldsymbol{Z}^\top \mathbf{1} = \mathbf{1} \right\},\ P_{ij} = \delta_{\pi(i),j}\ .$$

That is, each row and column of $\boldsymbol{P}$ sums to one. Equivalence holds in the sense, that if we stack the elements of the set X into a vector $\boldsymbol{x} = (x_1, \dots, x_n)^\top$, we obtain

$$\boldsymbol{P}\boldsymbol{x} = \left(x_{\pi(1)}, \dots, x_{\pi(n)}\right)^\top\ .$$

In words, a permutation is a rearrangement of elements in a set in terms of position. The corresponding permutation matrix swaps a vector's entries accordingly. We extend the term to injective maps.

Definition 7.2 (Subpermutation)**.** With $m, n \in \mathbb{N}, m \leq n$ we define a map $\pi : [m] \to [n]$ to be a subpermutation if it is injective. π can also be described by a binary subpermutation matrix, whose rows sum to 1 and whose columns contain at most a single 1. The space of subpermutation matrices of dimension $m \times n$ can hence be defined as

$$\mathbb{P}_{m,n} \coloneqq \left\{ \boldsymbol{Z} \in \{0,1\}^{m \times n} : \boldsymbol{Z}\mathbf{1}_n = \mathbf{1}_m,\ \boldsymbol{Z}^\top \mathbf{1}_m \preceq \mathbf{1}_n \right\}\ .$$

The QAP can be formulated as follows.

Definition 7.3 (QAP)**.** Given are a set of facilities $\mathcal{F} = \{p_1, \dots, p_m\}$ and a set of locations $\mathcal{L} = \{l_1, \dots, l_n\}$ $(m \leq n)$, along with a flow function $f : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$ between facilities and a distance function $d : \mathcal{L} \times \mathcal{L} \to \mathbb{R}$ between locations. We define the flow and distance matrices as

$$\boldsymbol{F} \coloneqq \left(f(p_i, p_j)\right)_{i,j=1}^{m}, \quad \boldsymbol{D} \coloneqq \left(d(l_i, l_j)\right)_{i,j=1}^{n}\ .$$

We formulate the quadratic assignment problem (QAP) as

$$\arg\min_{\pi} \sum_{i,j\in[m]} \boldsymbol{F}_{i,j}\boldsymbol{D}_{\pi(i),\pi(j)} \ ,$$

where $\pi : [m] \to [n]$ is a subpermutation. An equivalent formulation is given by the corresponding permutation matrices

$$\min_{\boldsymbol{P}\in\mathbb{P}_{m,n}} \operatorname{tr}\left(\boldsymbol{F}\boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^\top\right) \ .$$

For given flow and distance matrices $\boldsymbol{F}$ and $\boldsymbol{D}$, we define the cost function as

$$c\left(\boldsymbol{A},\boldsymbol{B}\right) \coloneqq \operatorname{tr}\left(\boldsymbol{F}\boldsymbol{A}\boldsymbol{D}\boldsymbol{B}^\top\right), \quad c\left(\boldsymbol{A}\right) \coloneqq c\left(\boldsymbol{A},\boldsymbol{A}\right) \ .$$

In the literature, assuming $m < n$ is denoted as *unbalanced* QAP, while a QAP is typically defined with $m = n$. In practice however, the number of facilities is often way smaller than the number of locations, as we will see in Section 7.4. We remark that the number of permutations $[n] \to [n]$ is $n!$, hence, it infeasible to use exhaustive search/brute force for finding an optimal permutation.

7.2.1 QUBO Formulation for the QAP

To obtain a QUBO formulation, we observe the that QAP is a specific form of QBP.

Proposition 7.1. *Let $\boldsymbol{F} \in \mathbb{R}^{m\times m}$, $\boldsymbol{D} \in \mathbb{R}^{n\times n}$, $m \leq n$, $\boldsymbol{W} \coloneqq \boldsymbol{F}\otimes\boldsymbol{D}$, $\boldsymbol{A} \coloneqq \boldsymbol{I}_m\otimes\boldsymbol{1}_n^\top$ and $\boldsymbol{B} \coloneqq \boldsymbol{1}_m^\top\otimes\boldsymbol{I}_n$. The following equivalence holds*

$$\begin{aligned} \min_{\boldsymbol{z}\in\mathbb{B}^{mn}} \ & \boldsymbol{z}^\top\boldsymbol{W}\boldsymbol{z} \\ \text{s.t. } & \boldsymbol{A}\boldsymbol{z} = \boldsymbol{1}_m \\ & \boldsymbol{B}\boldsymbol{z} \preceq \boldsymbol{1}_n \end{aligned} \quad \Leftrightarrow \quad \min_{\boldsymbol{P}\in\mathbb{P}_{m,n}} \operatorname{tr}\left(\boldsymbol{F}\boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^\top\right) \ .$$

Proof. For a binary $\boldsymbol{Z} \in \mathbb{B}^{m,n}$, let $\boldsymbol{z} = \operatorname{vec}(\boldsymbol{Z})$. Then

$$\boldsymbol{z}^\top\boldsymbol{W}\boldsymbol{z} = \boldsymbol{z}^\top(\boldsymbol{F}\otimes\boldsymbol{D})\boldsymbol{z} = \sum_{(i,j),(k,l)\in[m]\times[n]} Z_{ij}Z_{kl}F_{ij}D_{kl} = \operatorname{tr}\left(\boldsymbol{F}\boldsymbol{Z}\boldsymbol{D}\boldsymbol{Z}^\top\right) \ .$$

For the constraints, we remark that

$$\boldsymbol{A}\boldsymbol{z} = (\boldsymbol{I}_m\otimes\boldsymbol{1}_n)\boldsymbol{z} = \boldsymbol{Z}\boldsymbol{1}_n, \ \boldsymbol{B}\boldsymbol{z} = (\boldsymbol{1}_m^\top\otimes\boldsymbol{I}_n)\boldsymbol{z} = \boldsymbol{Z}^\top\boldsymbol{1}_m \ ,$$

which follows from standard matrix identities involving Kronecker products, matrix traces and the vec operator [252]. □

To obtain a QUBO formulation, we can leverage the equivalence in Proposition 7.1.

Proposition 7.2 (QAP-QUBO)**.** *Let $\boldsymbol{F} \in \mathbb{R}^{m\times m}$, $\boldsymbol{D} \in \mathbb{R}^{n\times n}$, $m \leq n$, $\boldsymbol{A} \coloneqq \boldsymbol{I}_m \otimes \boldsymbol{1}_n^\top$ and $\boldsymbol{B} \coloneqq \boldsymbol{1}_m^\top \otimes \boldsymbol{I}_n$. We can find a QUBO formulation for the* QAP *with $\boldsymbol{\lambda} \in \mathbb{R}_+^m$, $\boldsymbol{\mu} \in \mathbb{R}_+^n$, $\boldsymbol{\lambda}' \coloneqq (\sqrt{\lambda_i})_{i\in[m]}^\top$,*

$\boldsymbol{\mu}' \coloneqq (\sqrt{\mu_i})_{i\in[n]}^\top$, $\boldsymbol{A}' \coloneqq \boldsymbol{\lambda}'\mathbf{1}_m^\top \odot \boldsymbol{A}$, $\boldsymbol{B}' \coloneqq \boldsymbol{\mu}'\mathbf{1}_n^\top \odot \boldsymbol{B}$, $\mathbf{1}'_m \coloneqq \boldsymbol{\lambda}' \odot \mathbf{1}_m$ *and* $\mathbf{1}'_n \coloneqq \boldsymbol{\mu}' \odot \mathbf{1}_n$

$$\min_{\boldsymbol{z}\in\mathbb{B}^{mn}} \boldsymbol{z}^\top \boldsymbol{Q}\boldsymbol{z}, \quad \boldsymbol{Q} \coloneqq \boldsymbol{F}\otimes\boldsymbol{D} + \boldsymbol{A}'^\top\boldsymbol{A}' + \boldsymbol{B}'^\top\boldsymbol{B}' - \operatorname{diag}\left(2\boldsymbol{A}'^\top\mathbf{1}'_m + \boldsymbol{B}'^\top\mathbf{1}'_n\right) . \tag{7.1}$$

Proof. The QUBO formulation is obtained by applying Proposition 3.1 to Proposition 7.1 and using Proposition 6.3 for incorporating inequality constraints $\preceq \mathbf{1}$. □

For practical problems, the problem size mn can quickly get intractable for current quantum hardware, due to the limited qubit scale.

7.2.2 Logarithmic Encoding

A first try towards reducing the problem size is to use a different encoding scheme. We focus on the case $m = n$, the resulting state space size is then n^2. Our idea is to not use a permutation matrix but to use a matrix whose rows correspond to binary numbers represented as vectors. For this, we assume that $n = 2^k$, i.e., $k = \log_2 n$.

Definition 7.4. Let $k \in \mathbb{N}$, $n = 2^k$, $\pi : [n] \to [n]$ be a permutation, $\boldsymbol{P} \in \mathbb{P}_n$ its corresponding permutation matirx, $\boldsymbol{d} \coloneqq (2^{k-1}, \dots, 2^0)^\top$ and $\boldsymbol{n} \coloneqq (0, \dots, n-1)^\top$. We define the *binary representation matrix* $\boldsymbol{B}_\pi \in \{0,1\}^{n\times k}$ of π to be the matrix which maps the vector $\boldsymbol{d}$ to a permutation π of $\boldsymbol{n}$, i.e.,

$$\boldsymbol{B}_\pi\boldsymbol{d} = \boldsymbol{P}_\pi\boldsymbol{n} .$$

The rows of $\boldsymbol{B}_\pi$ correspond to binary numbers in $[n]$, since the entries in $\boldsymbol{n}$ are unique. We denote the set of all binary number matrices on $[n]$ as $\mathcal{B}_n$.

Example 5. As an example, set $n = 4$ and let $\pi(0) = 2$, $\pi(1) = 3$, $\pi(2) = 1$, $\pi(3) = 0$,

$$\boldsymbol{P}_\pi\boldsymbol{n} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \boldsymbol{B}_\pi\boldsymbol{d} . \tag{7.2}$$

With this representation our state space is reduced from n^2 to $n\log_2 n$. The number of permutations for n elements is $n!$ and thus for representing all permutations we need at least m bits such that

$$2^m \geq n! \;\Leftrightarrow\; m \geq \log_2(n!) \tag{7.3}$$

is fulfilled. With $\log_2(n!) = n\ln n - n + \Theta(\ln n)$, we can deduce that $\Omega(n\log_2 n)$ bits are needed for representing all permutations. Thus, our variable reduction is asymptotically optimal.

We have seen in opposition to Proposition 7.2 that a QUBO formulation for QAP can be obtained by incorporating constraints. However, the conditions for the binary representation matrix are not as trivial as for permutation matrices.

Proposition 7.3. *Let* $\boldsymbol{Z} = (\boldsymbol{z}_0 \;\cdots\; \boldsymbol{z}_{k-1}) \in \{0,1\}^{2^k\times k}$, $\boldsymbol{z}_i \in \{0,1\}^{2^k}$ *and let* $\boldsymbol{Z}_0$ *(*$\boldsymbol{Z}_1$*) be the submatrix of* $\boldsymbol{Z}$ *which arises from removing the first column and then only considering the row indices in which the*

first's column entry is 0 (1). Then for $k \geq 2$

$$\boldsymbol{Z} \in \mathcal{B}_{2^k} \;\Leftrightarrow\; \boldsymbol{z}_0^\top \boldsymbol{z}_0 = \boldsymbol{z}_0^\top \mathbf{1}_{2^k} = 2^{k-1} \;\wedge\; \boldsymbol{Z}_0, \boldsymbol{Z}_1 \in \mathcal{B}_{2^{k-1}}. \tag{7.4}$$

Proof. Without loss of generality we can assume that the rows of $\boldsymbol{Z}$ are ordered with respect to the ordering of binary numbers. This can be assumed, since both sides in Equation (7.4) do not get effected by applying row-wise permutations. We prove the claim using induction.
"$\Rightarrow$" Trivial.
"$\Leftarrow$" **IB**: Let $k = 1$. Then $\boldsymbol{Z} = (0, 1)^\top$, since $\boldsymbol{z}_0^\top \boldsymbol{z}_0 = 1$.
IS: With the ordering described above, we can write $\boldsymbol{Z}$ as

$$\boldsymbol{Z} = \begin{pmatrix} 0 & \cdots & 0 & 1 & \cdots & 1 \\ & \boldsymbol{Z}_0^\top & & & \boldsymbol{Z}_1^\top & \end{pmatrix}^\top, \tag{7.5}$$

with $\boldsymbol{Z}_0, \boldsymbol{Z}_1 \in \{0,1\}^{2^{k-1} \times k-1}$, since $\boldsymbol{z}_0$ has exactly 2^{k-1} ones and 2^{k-1} zeros. With the induction hypothesis the claim follows, since $\boldsymbol{Z}_0, \boldsymbol{Z}_1 \in \mathcal{B}_{2^{k-1}}$. □

The recursive condition in Equation (7.4) does not enlighten us directly how to find an equivalent quadratic formulation. We first setup a binary form of higher degree.

Proposition 7.4. *Let* $\boldsymbol{Z} = (\boldsymbol{z}_0 \;\cdots\; \boldsymbol{z}_{k-1}) \in \{0,1\}^{2^k \times k}$, $\boldsymbol{z}_i \in \{0,1\}^{2^k}$ *Then,* $\boldsymbol{B} \in \mathbb{B}^{2^k \times k}$ *if*

$$\boldsymbol{B} \in \operatorname{arg\,min}_{\boldsymbol{Z} \in \mathbb{B}^{2^k \times k}} \sum_{\mathcal{Z} \in \mathcal{Z}_l} \left[\mathbf{1}_{2^k}^\top \left(\bigodot_{\tilde{\boldsymbol{z}} \in \mathcal{Z}} \tilde{\boldsymbol{z}} \right) \left(\bigodot_{\tilde{\boldsymbol{z}}' \in \mathcal{Z}} \tilde{\boldsymbol{z}}' \right)^\top \mathbf{1}_{2^k} - 2^{k-l} \left(\bigodot_{\tilde{\boldsymbol{z}} \in \mathcal{Z}} \tilde{\boldsymbol{z}} \right)^\top \mathbf{1}_{2^k} \right], \tag{7.6}$$

where $\mathcal{Z}_l := \{\{\tilde{\boldsymbol{z}}_0, \ldots, \tilde{\boldsymbol{z}}_{l-1}, \boldsymbol{z}_l\} : \tilde{\boldsymbol{z}}_i \in \{\boldsymbol{z}_i, \mathbf{1}_{2^k} - \boldsymbol{z}_i\},\ i \in [l]\}$.

Proof. Note that the matrices $\boldsymbol{Z}_0, \boldsymbol{Z}_1$ can be obtained by entry-wise multiplying all columns but the first column by $\mathbf{1}_{2^k} - \boldsymbol{z}_0$ and $\boldsymbol{z}_0$, respectively. We can then use Equation (7.4) iteratively, such that all submatrices fulfill this condition. For the l-th column of $\boldsymbol{Z}$, we thus get the following conditions:

$$\left(\bigodot_{\tilde{\boldsymbol{z}} \in \mathcal{Z}} \tilde{\boldsymbol{z}} \right)^\top \mathbf{1}_{2^k} = 2^{k-l},\ \mathcal{Z} \in \mathcal{Z}_l\ .$$

This leads to 2^l conditions for the $l+1$-th column. We can write

$$\begin{aligned}
&\left(\bigodot_{\tilde{\boldsymbol{z}}\in\mathcal{Z}} \tilde{\boldsymbol{z}}\right)^\top \mathbf{1}_{2^k} = 2^{k-l-1},\ \mathcal{Z}\in\mathcal{Z}_l \\
\Leftrightarrow &\left[\left(\bigodot_{\tilde{\boldsymbol{z}}\in\mathcal{Z}} \tilde{\boldsymbol{z}}\right)^\top \mathbf{1}_{2^k} - 2^{k-l-1}\right]^2 = 0,\ \mathcal{Z}\in\mathcal{Z}_l \\
\Leftrightarrow &\min_{\boldsymbol{Z}\in\mathbb{B}^{2^k\times k}} \sum_{\mathcal{Z}\in\mathcal{Z}_l} \left[\left(\bigodot_{\tilde{\boldsymbol{z}}\in\mathcal{Z}} \tilde{\boldsymbol{z}}\right)^\top \mathbf{1}_{2^k} - 2^{k-l-1}\right]^2 \\
\Leftrightarrow &\min_{\boldsymbol{Z}\in\mathbb{B}^{2^k\times k}} \sum_{\mathcal{Z}\in\mathcal{Z}_l} \left[\mathbf{1}_{2^k}^\top \left(\bigodot_{\tilde{\boldsymbol{z}}\in\mathcal{Z}} \tilde{\boldsymbol{z}}\right)\left(\bigodot_{\tilde{\boldsymbol{z}}'\in\mathcal{Z}} \tilde{\boldsymbol{z}}'\right)^\top \mathbf{1}_{2^k} - 2^{k-l}\left(\bigodot_{\tilde{\boldsymbol{z}}\in\mathcal{Z}} \tilde{\boldsymbol{z}}\right)^\top \mathbf{1}_{2^k}\right] .
\end{aligned}$$

□

The optimization problem in Equation (7.6) is not quadratic and it is thus not straightforward to obtain a QUBO formulation. A higher-degree pseudo boolean function can be quadratized by using penalty terms [253]. Specifically, a product of two binary variables $z_i z_j$ can be replaced by a single binary variable $z_i z_j = z_{ij}$ by adding

$$M(z_i z_j - 2z_i z_{ij} - 2z_j z_{ij} + 3z_{ij}) ,$$

to the objective function, where $M > 0$ is a suitable penalty term. Using this substitution, we need $\mathcal{O}(n^2)$ auxiliary variables, rendering the official size reduction useless. This could be circumvented by using more sophisticated quadratization techniques [254, 255]. In the best case however, we still need at least $n\log_2(n)$ qubits, which is out of reach with a large n. Thus, we examine a different technique suited for permutation-based optimization, whit adaptable subproblem size.

7.3 Cyclic Expansion

The above issues can be overcome by not considering a single QUBO formulation but a series of QUBOs. Compared to the methods discussed in Chapters 5 and 6 we do not consider the optimization of subproblems, but take a slightly different approach. That is, we use a variant of the α-expansion algorithm [39, 197]. The original α-expansion is a graph-cut-based method that efficiently finds approximate solutions for problems where the objective function involves pairwise interactions between variables. The key idea is to iteratively improve a current labeling by allowing nodes to either keep their current label or switch to a given label α. We expand this approach by considering permutation-based optimization—the QAP problem in particular. The idea is that instead of optimizing over all permutation matrices at once, an iterative optimization over cyclic permutations is carried out which converges towards the original optimization. For the upcoming sections we assume that $m = n$, the case $m < n$ follows analogously.

Informally, the cyclic expansion algorithm works as follows:

1. Initialize permutation matrix $\boldsymbol{P} \in \mathbb{P}_n$,
2. Choose a set of *simple* permutation matrices $\mathcal{C} \subset \mathbb{P}_n$ with $|\mathcal{C}| = k < n$,
3. Solve a QUBO of size k to decide which permutation in $\mathcal{C}$ to apply to $\boldsymbol{P}$,
4. Update $\boldsymbol{P}$ with the chosen permutation,
5. Repeat steps 2-4 until convergence of cost $c(\boldsymbol{P})$.

In what follows, we define all necessary terms and provide a detailed description of Algorithm 8, also depicted in Figure 7.1. As *simple* permutation matrices we use cyclic permutations, even the smallest possible cyclic permutations, so called 2-cycles.

Definition 7.5 (**2**-Cycle)**.** Let $\boldsymbol{C} \in \mathbb{P}_n$ be a permutation matrix. $\boldsymbol{C}$ is a 2-cycle if

$$\exists i, j \in [n], i \neq j : C_{ij} = C_{ji} = 1 \text{and} \quad \forall l \in [n], l \notin \{i, j\} : C_{ll} = 1 \ .$$

We denote the set of 2-cycles with $\mathbb{P}_n^{(2)}$.

A 2-cycle is also called a *transposition*. We remark that any permutation matrix $\boldsymbol{P} \in \mathbb{P}_n$ can be written as a product of 2-cycles,

$$\forall \boldsymbol{P} \in \mathbb{P}_n : \exists \{C_1, \ldots, C_s\} \subset \mathbb{P}_n^{(2)} : \boldsymbol{P} = \prod_{i=1}^{s} \boldsymbol{C}_i \ . \tag{7.8}$$

Instead of optimizing over all 2-cycles the idea is to iteratively consider fixed subsets of $\mathbb{P}_n^{(2)}$.

Definition 7.6. Let $\mathcal{C} = \{\boldsymbol{C}_1, \ldots, \boldsymbol{C}_s\} \subset \mathbb{P}_n^{(2)}$ be a set of 2-cycles and let $\boldsymbol{\alpha} \in \mathbb{B}^s$, $s \in \mathbb{N}$. For $\boldsymbol{P} \in \mathbb{P}_n$ we define

$$g\left(\boldsymbol{P}, \boldsymbol{\alpha}, \mathcal{C}\right) \coloneqq \left(\prod_{i=1}^{s} \boldsymbol{C}_i^{\alpha_i}\right) \boldsymbol{P} = \boldsymbol{C}_1^{\alpha_1} \cdots \boldsymbol{C}_s^{\alpha_s} \boldsymbol{P} \ , \tag{7.9}$$

with $\boldsymbol{C}_i^0 \coloneqq \boldsymbol{I}_n$, $\boldsymbol{C}_i^1 \coloneqq \boldsymbol{C}_{\boldsymbol{i}}$. In words, the vector $\boldsymbol{\alpha}$ indicates which cycle in $\mathcal{C}$ should be applied to $\boldsymbol{P}$.

For a given $\boldsymbol{P} \in \mathbb{P}_n$ the following objective is optimized in each iteration

$$\underset{\boldsymbol{\alpha} \in \mathbb{B}^s}{\arg\min} \ c\left(g\left(\boldsymbol{P}, \boldsymbol{\alpha}, \mathcal{C}\right)\right) \ . \tag{7.10}$$

However, Equation (7.10) is not in QUBO form and can thus not be directly solved on actual quantum hardware. To overcome this issue, we only consider disjoint 2-cycles.

Definition 7.7. Let $\boldsymbol{C}, \boldsymbol{C}' \in \mathbb{P}_n^{(2)}$. $\boldsymbol{C}$ and $\boldsymbol{C}'$ are disjoint if

$$\left(C_{ii} = 0 \Rightarrow C'_{ii} = 1\right) \wedge \left(C'_{ii} = 0 \Rightarrow C_{ii} = 1\right) \ ,$$

which leads to commutativity, i.e., $\boldsymbol{C}\boldsymbol{C}' = \boldsymbol{C}'\boldsymbol{C}$. We call a set $\mathcal{C} \subset \mathbb{P}_n^{(2)}$ disjoint if all elements are pairwise disjoint.

Sets of disjoint 2-cycles have a large expressive power in terms of covering the whole permutation space.

Proposition 7.5. *Given a permutation matrix $\boldsymbol{P}$, there exist two sets $\mathcal{C}, \mathcal{C}' \subset \mathbb{P}_n^{(2)}$ of disjoint 2-cycles such that*

$$\boldsymbol{P} = \boldsymbol{L}\boldsymbol{R}, \quad \boldsymbol{L} := \prod_{\boldsymbol{C} \in \mathcal{C}} \boldsymbol{C}, \quad \boldsymbol{R} := \prod_{\boldsymbol{C}' \in \mathcal{C}'} \boldsymbol{C}' \, .$$

Proof. See [197]. □

Assuming disjoint 2-cycles we obtain the following result.

Lemma 7.1. *Assume that $\mathcal{C} = \{\boldsymbol{C}_1, \ldots, \boldsymbol{C}_s\} \subset \mathbb{P}_n^{(2)}$ is a disjoint set of 2-cycles and let $\boldsymbol{P} \in \mathbb{P}_n$ be a permutation matrix. Then, the following identity holds*

$$g\left(\boldsymbol{P}, \boldsymbol{\alpha}, \mathcal{C}\right) = \boldsymbol{P} + \sum_{i=1}^{s} \alpha_i \tilde{\boldsymbol{C}}_i, \quad \tilde{\boldsymbol{C}}_i := \left(\boldsymbol{C}_i - \boldsymbol{I}_n\right) \boldsymbol{P} \, . \tag{7.11}$$

Proof. We prove the statement by induction. For $s = 1$ Equation (7.9) reduces to

$$\boldsymbol{C}^{\alpha} \boldsymbol{P} = (1 - \alpha) \, \boldsymbol{P} + \alpha \boldsymbol{C} \boldsymbol{P} = \boldsymbol{P} + \alpha \left(\boldsymbol{C} - \boldsymbol{I}_n\right) \boldsymbol{P} \, ,$$

leading to Equation (7.11). Multiplying the inverse of $\boldsymbol{P}$ to the right, we obtain

$$\boldsymbol{C}^{\alpha} = \boldsymbol{I}_n + \alpha \left(\boldsymbol{C} - \boldsymbol{I}_n\right) \, .$$

Now, consider s and assume that Equation (7.11) holds for $s - 1$. Then

$$\begin{aligned}
&\left(\prod_{i=1}^{s} \boldsymbol{C}_i^{\alpha_i}\right) = \left(\prod_{i=1}^{s-1} \boldsymbol{C}_i^{\alpha_i}\right) \boldsymbol{C}_s^{\alpha_s} \\
&= \left(\boldsymbol{I}_n + \sum_{i=1}^{s-1} \alpha_i \left(\boldsymbol{C}_i - \boldsymbol{I}_n\right)\right) \left(\boldsymbol{I}_n + \alpha_s \left(\boldsymbol{C}_s - \boldsymbol{I}_n\right)\right) \\
&= \left(\boldsymbol{I}_n + \sum_{i=1}^{s} \alpha_i \left(\boldsymbol{C}_i - \boldsymbol{I}_n\right)\right) \left(\sum_{i=1}^{s} \alpha_i \alpha_s \left(\boldsymbol{C}_i - \boldsymbol{I}_n\right) \left(\boldsymbol{C}_s - \boldsymbol{I}_n\right)\right) \, ,
\end{aligned}$$

and it remains to show that $\left(\boldsymbol{C}_i - \boldsymbol{I}_n\right) \left(\boldsymbol{C}_s - \boldsymbol{I}_n\right) = \boldsymbol{0}\boldsymbol{0}^{\top}$. Since all $\boldsymbol{C}_i$ are 2-cycles, $\boldsymbol{C}_i - \boldsymbol{I}_n$ only has 4 non-zeros entries and since $\boldsymbol{C}_i$ and $\boldsymbol{C}_s$ are disjoint, they have these entries in different rows/columns. This leads to their product being equal to the matrix consisting only of zeros. □

Inserting Equation (7.11) into Equation (7.10) leads to the following QUBO.

Theorem 7.1 (CyclicExpansion-QUBO)**.** *Assume that $\mathcal{C} = \{\boldsymbol{C}_1, \ldots, \boldsymbol{C}_s\} \subset \mathbb{P}_n^{(2)}$ is a disjoint set of 2-cycles, $\boldsymbol{P} \in \mathbb{P}_n$. A QUBO formulation equivalent to Equation (7.10) is given by*

$$\min_{\boldsymbol{\alpha} \in \mathbb{B}^s} \boldsymbol{\alpha}^{\top} \boldsymbol{Q} \boldsymbol{\alpha}, \quad Q_{ij} := \begin{cases} c\left(\tilde{\boldsymbol{C}}_i, \tilde{\boldsymbol{C}}_j\right), & \text{if } i \neq j \, , \\ c\left(\tilde{\boldsymbol{C}}_i\right) + c\left(\tilde{\boldsymbol{C}}_i, \boldsymbol{P}\right) + c\left(\boldsymbol{P}, \tilde{\boldsymbol{C}}_i\right), & \text{else} \, . \end{cases} \tag{7.12}$$

Algorithm 8 CYCLICEXPANSION Algorithm

Input: $\boldsymbol{F} \in \mathbb{R}^{m \times m}$, $\boldsymbol{D} \in \mathbb{R}^{n \times n}$, $k \leq m$, $k_u \leq n - k$
Output: subpermutation matrix $\boldsymbol{P} \in \mathbb{P}_{m,n}$ optimizing $c(\boldsymbol{P})$

```
Initialize P ∈ P_{m,n}
repeat
    Choose indices I, J, |I| = k, |J| = k_u (Section 7.4.2)
    Construct matrix W(I, J) (Section 7.4.2)
    repeat
        Choose a random set of 2-cycles ℂ (Section 7.4.2)
        Calculate Q(I, J) from W(I, J) (Theorem 7.1)
        α* ← arg min_α α^⊤ Q(I, J) α                    ▷ QC
        P ← g(P, α*, ℂ) (Lemma 7.1)
    until Every 2-cycle occured in one set
until A convergence criterium is met
```

Proof. Since tr and matrix multiplication are linear functions, c is bilinear and we obtain

$$
\begin{aligned}
&\min_{\boldsymbol{\alpha} \in \mathbb{B}^s} \quad c\left(g\left(\boldsymbol{P}, \boldsymbol{\alpha}, \mathcal{C}\right)\right) \\
=\ &\min_{\boldsymbol{\alpha} \in \mathbb{B}^s} \quad c\left(\boldsymbol{P} + \sum_{i=1}^{s} \alpha_i \tilde{\boldsymbol{C}}_i, \boldsymbol{P} + \sum_{j=1}^{s} \alpha_j \tilde{\boldsymbol{C}}_j\right) \\
=\ &\min_{\boldsymbol{\alpha} \in \mathbb{B}^s} \quad c\left(\boldsymbol{P}\right) + \sum_{i,j=1}^{s} \alpha_i \alpha_j c\left(\tilde{\boldsymbol{C}}_i, \tilde{\boldsymbol{C}}_j\right) + \sum_{i=1}^{s} \alpha_i c\left(\tilde{\boldsymbol{C}}_i, \boldsymbol{P}\right) + \sum_{j=1}^{s} \alpha_j c\left(\boldsymbol{P}, \tilde{\boldsymbol{C}}_j\right) \\
=\ &\min_{\boldsymbol{\alpha} \in \mathbb{B}^s} \quad \boldsymbol{\alpha}^\top \boldsymbol{Q} \boldsymbol{\alpha}\ ,
\end{aligned}
$$

where $\boldsymbol{Q}$ is defined as in Equation (7.12). □

We observe that for a set with n elements, the largest possible set of disjoint 2-cycles has $\lfloor n/2 \rfloor$ elements. Therefore, the dimension of the QUBO problem in Equation (7.12) is way smaller than the size of the original QUBO in Proposition 7.2 ($s \leq \lfloor n/2 \rfloor \ll m(n+1)$).

The overall iterative method is outlined in Algorithm 8: Given a permutation matrix $\boldsymbol{P} \in \mathbb{P}_n$, we iteratively choose sets of random disjoint cycles and optimize Equation (7.12). This gives us a binary vector $\boldsymbol{\alpha}$, indicating which cycle should be applied to our current permutation matrix. After updating $\boldsymbol{P}$, the procedure is repeated until convergence. The specifics for Algorithm 8 are elaborated in the next section.

7.4 Application: FPGA-Placement

Logic optimization, placing and routing are fundamental and the most time-consuming steps in the field of chip design for both ASICs and FPGAs [256]. The number of transistors and logic gates on a single chip is increasing more and more, leading to the mentioned processes consuming more and more time.

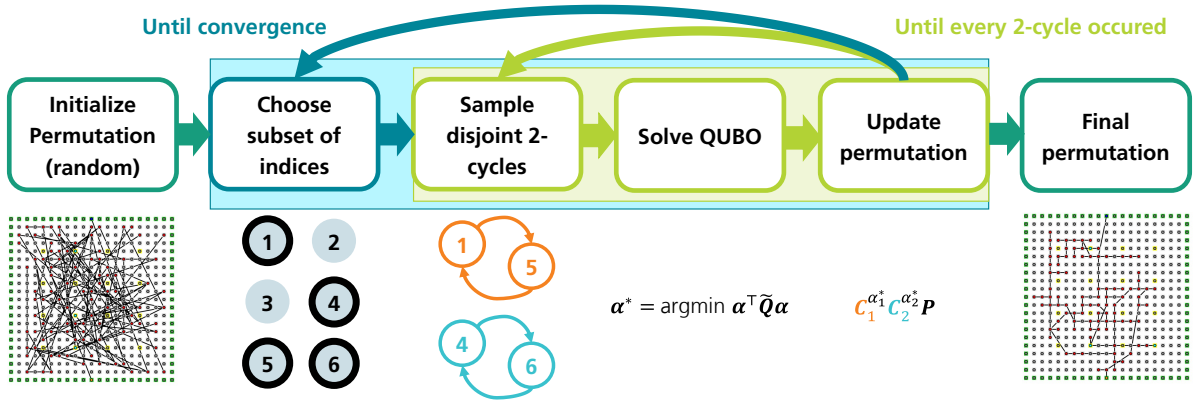


Figure 7.2: Flow chart of the CYCLICEXPANSION (Algorithm 8): Given an initial permutation (random), we choose a subproblem and iteratively sample random disjoint cycles, which are used to formulate a QUBO problem. This QUBO formulation is solved with quantum annealing, giving us a binary vector α^*, indicating which cycle should be applied to the current permutation. If every cycle occurred, we choose a new subproblem and repeat this procedure until convergence.

This limits the speed of development cycles, which is an issue of productivity but can also be a security issue, since faster development cycles for cryptography related algorithms allow for improved security analysis.

Here, we focus on the placement step, in which we aim to find an ideal placement of functional blocks on the chip. The advantages of a good placement are twofold: On the one hand, minimizing the physical distance between connected elements leads to shorter wire lengths and therefore a higher maximum clock rate. On the other hand, a good placement can lead to a faster routing process, i.e., the routing algorithm of the connections between elements finding a good solution in fewer iterations and less time. Since the placement itself is an increasingly time-consuming step, decreasing the runtime of the placement algorithm while maintaining a high solution quality is of great interest.

Taking a closer look at the math behind the placement in the floor-planning case, we find that it is equivalent to the QAP [237, 238]. The goal of the QAP is to assign each given element to a unique location, minimizing a given cost function. In chip design, that cost function can be the total wire length between connected units given by a placement on the chip. Minimizing that function leads to a shorter maximum wire length and with that the possibility for a higher maximum clock rate. The QAP is an NP-hard combinatorial optimization problem, and moreover, one of the hardest in this class, since there is no approximation algorithm for producing a suboptimal solution with guarantees in polynomial time [257].

While real-world quantum devices suffer from a series of technical limitations, there is theoretical evidence that hard combinatorial problems can be solved exactly via AQC. We leverage these theoretical insights and describe the first algorithm for placement of functional blocks (e.g., Lookup Tables (LUT), BlockRAMs (BRAM), Digital Signal Processing (DSP)) on an FPGA, based on solving a QAP via QC. That is, we apply our developed CYCLICEXPANSION algorithm to FPGA-placement, with an overview given in Figure 7.2.

7.4.1 FPGA-Placement

We illustrate the FPGA development procedure on the example of the open-source tool *nextpnr* [258], which is used for logic optimization, placement and routing. In the first step, nextpnr transforms the logic elements (LUTs, BRAMs, DSPs) from the logic optimization into elements which are actually available on the specific chip. The number of elements needed to be placed on the chip, as well as the neccessary conncetions between the elements, is then known exactly. After that step, nextpnr generates an initial placement guess, which is used as a seed for a simulated annealing approach. The goal of the placement is to determine an optimal spatial arrangement of these blocks on an FPGA to minimize communication delays and enhance performance. Following the placement, the routing is done by an A* algorithm with a rip-up and reroute strategy. The routing time is significantly affected by the quality of the placement, meaning a better placement can increase the routing speed. It is well known that the placement problem can be formulated as an unbalanced QAP. We now recap this formulation, since our construction in Section 7.3 relies on it to transform the placement problem into a series of QUBO problems.

In FPGA-placement the goal is to place m functional blocks into n physical slots on the FPGA chip grid such that the total wire length is minimized, with $m \leq n$. The matrix $\boldsymbol{F}$ indicates how two functional blocks are connected in the given net list and the distance matrix $\boldsymbol{D}$ indicates the distances between different locations on the chip. However, in the classical QAP framework, we need the two matrices to have the same dimensionality.

Using this definition, we can formalize the placement problem as a QAP by introducing a new matrix $\boldsymbol{F}' \in \mathbb{R}^{n\times n}$, which is 0 everywhere except for its $m \times m$ upper block matrix, i.e., $\boldsymbol{F}'_{[m],[m]} = \boldsymbol{F}$. Descriptively, we introduce $n - m$ "dummy" functional blocks which are not connected to any other unit. The placement objective can now be written as

$$\min_{\boldsymbol{P}\in\mathbb{P}_n} \operatorname{tr}\left(\boldsymbol{F}'\boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^\top\right) . \tag{7.13}$$

Even though it is a common way for obtaining a QAP formulation, inserting $n - m$ dummy elements leads to a large amount of redundancy and high dimensionality, especially if $m \ll n$. We can overcome this issue by considering subpermutations, discussed in Section 7.2.

Even though with Proposition 7.2, we have a quantum-compatible problem formulation for the QAP at hand, we remark that our problem dimension is mn. That is, for solving an FPGA-placement problem with m functional blocks and n grid locations, we need nm qubits, which is beyond capabilities of current (and upcoming) quantum hardware. Furthermore, choosing the penalty parameters $\boldsymbol{\lambda}, \boldsymbol{\mu}$ in Proposition 7.2 maintaining equivalence while also having preferable conditioning for quantum hardware is tedious and error prone. In [145], coarse upper bounds are provided for these parameters. Moreover, constraints on the permutation space can not easily be integrated into such a formulation. However, this is of great importance in FPGA-placement since we have to take into account that the types of every functional block and the corresponding placement location have to match. For example, it is impossible to place a LUT onto an IO location (cf. Figure 7.3). We thus investigate the performance of the CyclicExpansion-algorithm developed in Section 7.3 for FPGA-placement.

7.4.2 Implementation Details

We give an overview of the implementation details of our proposed algorithm in Algorithm 8.

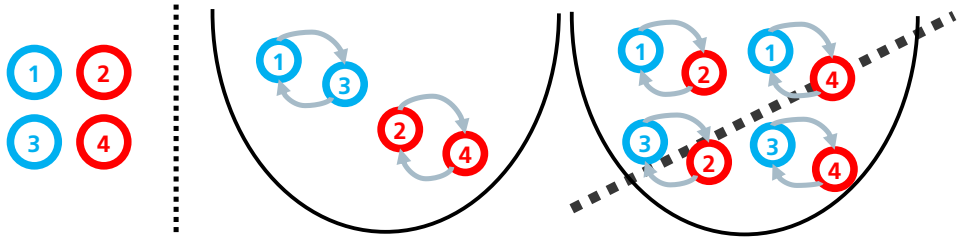


Figure 7.3: Illustration of allowed 2-cycles. Shown are the current placement (left) with corresponding cell types (cyan and red) and an example of legal and illegal 2-cycles (right).

Initialization Since our proposed method works iteratively, any given initial solution can be incorporated easily. Either we can start off with a random subpermutation or something more elaborated like analytical or force-directed placement [238].

Choosing Indices Instead of optimizing over the whole index set $[m]$ in each iteration we can reduce the problem size by considering an index set $\mathcal{I} \subset [m]$ of size k. These indices can be chosen randomly but having a deep understanding of the underlying problem setting, one could use a more informative approach. For example, we could choose the indices depending on the impact of the overall cost, i.e.,

$$\max_{\mathcal{I}\subset[m],|\mathcal{I}|=k} \sum_{i\in\mathcal{I}} \sum_{j\in[n]} F_{i,j} D_{\pi(i),\pi(j)} \ . \tag{7.14}$$

Intuitively, it makes sense to greedily permute the currently worst performing indices. However, one might get stuck in a local optimum too early. Both methods (random and greedy) are investigated later on in Section 7.5.

Even though we now have a problem dimension only dependent on k and not the number of locations n, this approach is not yet guaranteed to converge towards an optimal solution. If we only consider index sets $\mathcal{I} \subset [m]$ we stick to permute the initial subpermutation and thus concentrate on a fixed set of m locations. To prevent this, in each iteration, we also sample a set $\mathcal{J} \subset [n]$ of $k_u \leq k$ unbound locations, i.e., locations which are not assigned to a functional block. We sample each unbound location with a probability proportional to the distance to the nearest neighbor in the set of bound locations. With this method, we also explore the set of unbound locations and can place the given functional blocks on the whole chip grid.

Choosing Cycles For fixed numbers of indices $k, k_u \leq m$, we iteratively sample a set of disjoint 2-cycles and optimize the current permutation until every 2-cycle has occurred in the sampling process (Lines 5-10). The question arises on which indices these cycles should be sampled. Considering an index set $\mathcal{I} = \{i_1, \ldots, i_k\} \subset [m]$ we can compute the subpermuted index set under the subpermutation π as $\mathcal{I}_\pi := \{\pi(i_1), \ldots, \pi(i_k)\} \subset [n]$. We first sample k_u disjoint 2-cycles which map $\{i_1, \ldots, i_{k_u}\}$ to $\mathcal{J}$. Secondly, we sample $\lfloor (k-k_u)/2 \rfloor$ disjoint 2-cycles which map $\{i_{k_u+1}, \ldots, i_k\}$ to $\{\pi(i_{k_u+1}), \ldots, \pi(i_k)\}$. Since there are restrictions on which functional block can be mapped to which chip location (e.g. a LUT cannot be placed on an IO cell), one cannot simply sample these cycles arbitrarily between chosen indices (see Figure 7.3). However, this does not pose a problem for this framework, because these constraints can be integrated into the sampling process. The total number of 2-cycles is thus $s = k_u + \lfloor (k-k_u)/2 \rfloor$, which is only dependent on the freely selectable index set sizes

k and k_u. Since the QUBO dimension corresponds exactly to the number of considered disjoint cycles, we can conveniently adapt the problem to the available hardware size, either for real quantum annealers or classical digital annealing QUBO solvers. However, there is a trade-off between problem size and performance, which will be investigated later on in Section 7.5.

Constructing Clamped Matrix It remains to clarify how the matrix $\boldsymbol{W}(\mathcal{I}, \mathcal{J})$ from Algorithm 8 is constructed. One way would be to precompute the cost matrix $\boldsymbol{W} = \boldsymbol{F} \otimes \boldsymbol{D} \in \mathbb{R}^{mn \times mn}$ and then using standard methods for reducing the QUBO size with fixed variables. Since real-world algorithms to be implemented on an FPGA-chip can contain up to millions of functional blocks and chip locations, the size of the cost matrix $\boldsymbol{W} \in \mathbb{R}^{mn \times mn}$ can get infeasible to hold the precomputed matrix in memory. We resolve this issue by exploiting the tensor product-like structure of $\boldsymbol{W} = \boldsymbol{F} \otimes \boldsymbol{D}$.

Definition 7.8. Let $m, n \in \mathbb{N}$, $m \leq n$, $\mathcal{I} \subset [m]$ and $\pi : [m] \to [n]$ be a subpermutation with corresponding subpermutation matrix $\boldsymbol{P} \in \mathbb{P}_{m,n}$. Define $\mathcal{I}^c := [m] \setminus \mathcal{I}$ and let $\mathcal{I}_\pi, \mathcal{I}_\pi^c \subset [n]$ be the sets created by applying the underlying subpermutation π of $\boldsymbol{P}$ to $\mathcal{I}, \mathcal{I}^c$.

Proposition 7.6. *Let $m, n \in \mathbb{N}$, $m \leq n$, $\pi : [m] \to [n]$ be a subpermutation with corresponding subpermutation matrix $\boldsymbol{P} \in \mathbb{P}_{m,n}$, $\boldsymbol{F} \in \mathbb{R}^{m \times m}$, $\boldsymbol{D} \in \mathbb{R}^{n \times n}$, $\boldsymbol{W} = \boldsymbol{F} \otimes \boldsymbol{D}$, $\mathcal{I} \subset [m]$, with $|\mathcal{I}| = k$ and $\mathcal{J} \subset [n]$ with $|\mathcal{J}| = k_u$. Then*

$$\min_{\boldsymbol{z}(\mathcal{I},\mathcal{J}) \in \{0,1\}^{k(k+k_u)}} \boldsymbol{z}^\top \boldsymbol{W} \boldsymbol{z} = \min_{\boldsymbol{z}(\mathcal{I},\mathcal{J}) \in \{0,1\}^{k(k+k_u)}} \left(\boldsymbol{z}(\mathcal{I}, \mathcal{J})\right)^\top \boldsymbol{W}(\mathcal{I}, \mathcal{J}) \, \boldsymbol{z}(\mathcal{I}, \mathcal{J}) \ ,$$

with $\boldsymbol{z}(\mathcal{I}, \mathcal{J}) := \mathrm{vec}(\boldsymbol{P}_{\mathcal{I},\mathcal{I}'_\pi})$, $\mathcal{I}'_\pi := \mathcal{I}_\pi \cup \mathcal{J}$, $\boldsymbol{z} = \mathrm{vec}(\boldsymbol{P})$ and

$$\boldsymbol{W}(\mathcal{I}, \mathcal{J}) := \boldsymbol{F}_{\mathcal{I}} \otimes \boldsymbol{D}_{\mathcal{I}'_\pi} + \mathrm{diag}\left(\mathrm{vec}\left(\boldsymbol{F}_{\mathcal{I},\mathcal{I}^c} \boldsymbol{D}_{\mathcal{I}_\pi^c,\mathcal{I}'_\pi} + \boldsymbol{F}_{\mathcal{I}^c,\mathcal{I}}^\top \boldsymbol{D}_{\mathcal{I}_\pi^c,\mathcal{I}'_\pi}\right)\right) \ .$$

With having clarified all steps of Algorithm 8, we can examine the behavior of this algorithm.

7.5 Experimental Evaluation

We conduct experiments with a fictional FPGA architecture for analyzing the behavior of our proposed algorithm. We choose this as a generic minimum baseline for all FPGA architectures, ignoring implementation details like grouping into slices, carry chains etc, which might be vendor specific and thus not translate easily to other FPGAs. In this architecture, we assume that every LUT has an adjacent register, so that its usage is irrelevant to the placement process and can be ignored. We consider only the data path, i.e., ignore clock net routing and control signals like reset and clock enable. However, this information can be integrated into $\boldsymbol{F}$ and $\boldsymbol{D}$.

The fictional FPGA architecture contains three different cell types: IO cells, BRAM cells and LUT cells. The legend for upcoming plots is indicated in Figure 7.4. For the upcoming experiments we assume an FPGA chip which consists of 21×21 cells. It contains IO cells at the border and 16 BRAM cells distributed uniformly over the gird, with the rest being LUT cells (see e.g. Figure 7.7). We are thus faced with $n = 21^2 = 441$ locations.

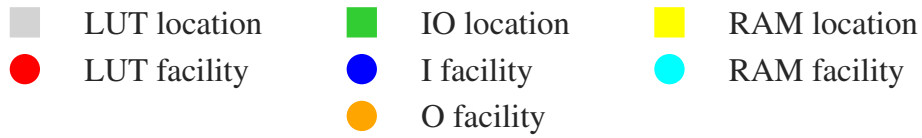
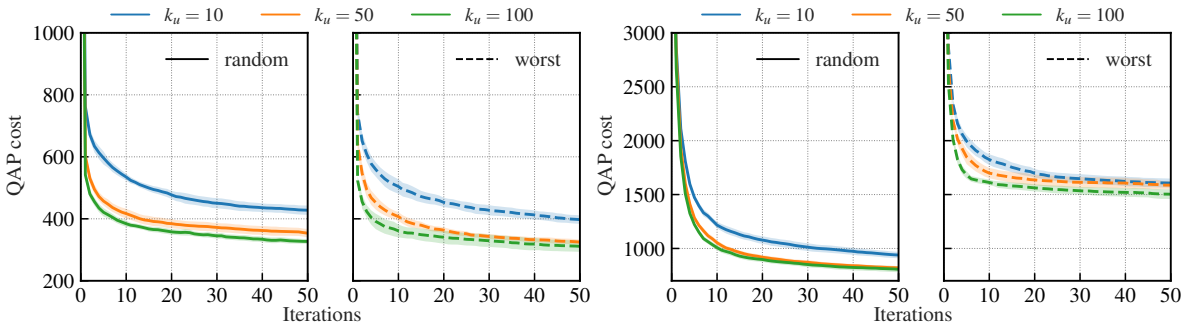


Figure 7.4: Different cell types for our fictional FPGA architecture along with plotted colors.



(a) $m = 100$. (b) $m = 200$.

Figure 7.5: Depicting the effect of varying k_u when k is fixed to a certain value, comparing choosing random subproblems in Line Algorithm 8 with choosing worst performing indices Equation (7.14). Here, $k = 100$ and the QAP cost is depicted over 50 iterations used in the CYCLICEXPANSION. We compare the costs for 10 randomly generated problems with a problem size of 100 (a) with 10 problems of size 200 (b).

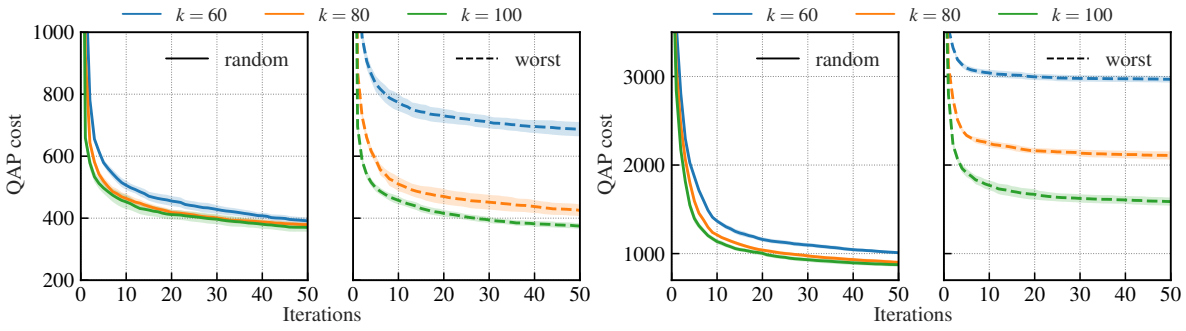
7.5.1 Generic Examples

For examining the behavior of Algorithm 8, we sample 10 different problem instances with $m = 100$ and $m = 200$ facilities, respectively. For every instance we assume two IO cells, imitating a single input and a single output cell. The rest of the cell types are randomly sampled corresponding to the ratio of the underlying architecture.

We compare the performance of Algorithm 8 with solving the QUBO given in Equation (7.1) using different QUBO solvers. As a classical software solver we use a simulated annealing (SA) implemented in the python software package D-Wave Ocean[2] with default parameters. As a second classical solver, we utilize a QUBO hardware solver, which is denoted as digital annealing (DA). Similar to QA, DA is standalone but is not based on quantum technology and uses classical algorithms. One can set up the running time/annealing time of this device and we henceforth set this time to $0.1\ s$ which is equivalent to evaluating $\approx 160k$ candidate solutions. Thirdly, we use a real quantum annealer (QA), namely a D-Wave Advantage System 5.4 with 5614 qubits and 40,050 couplers, fixing the annealing time to $40\ \mu s$ and taking the best out of 100 reads. Since our algorithm CYCLICEXPANSION contains random decisions, such as choosing a set of cycles in Line 6, we plot the average performance over 10 runs and indicate the 95%-confidence intervals.

We start with depicting the performance of CYCLICEXPANSION over 50 iterations in terms of the QAP cost in Figure 7.5, varying the number of chosen unbound indices k_u. We fix $k = 100$ and compare $k_u \in \{10, 50, 100\}$ using the SA solver, with the performance being averaged over the 10 generated instances for $m = 100$ and $m = 200$. Moreover, the impact of different methods for

[2] https://docs.ocean.dwavesys.com/en/stable (last accessed September 19, 2025)



(a) $m = 100$. (b) $m = 200$.

Figure 7.6: Depicting the effect of varying k when k_u is fixed to a certain value, comparing choosing random subproblems in Line Algorithm 8 with choosing worst performing indices Equation (7.14). Here, $k_u = 30$ and the QAP cost is depicted over 50 iterations used in the CYCLICEXPANSION. We compare the costs for 10 randomly generated problems with a problem size of 100 (a) with 10 problems of size 200 (b).

choosing subproblems in Line 3 is indicated, comparing random sampling with worst performing indices Equation (7.14).

We observe that the QAP cost decreases with every iteration of the algorithm. For $m = 100$ the random indices choosing method performs similar to the method of worst indices, contrary to the case $m = 200$. The larger the dimension of our problem gets, considering only the worst indices can lead to fast convergence to local optima, leading to an overall worse placement in the end. Furthermore, we can see that with an increasing number of unbound variables k_u, the performance of the CYCLICEXPANSION increases, since the space of unbound cells is more thoroughly explored. However, increasing this parameter k_u also leads to a larger QUBO size (Equation (7.12)), leading to a trade-off between problem size and performance for a fixed number of iterations.

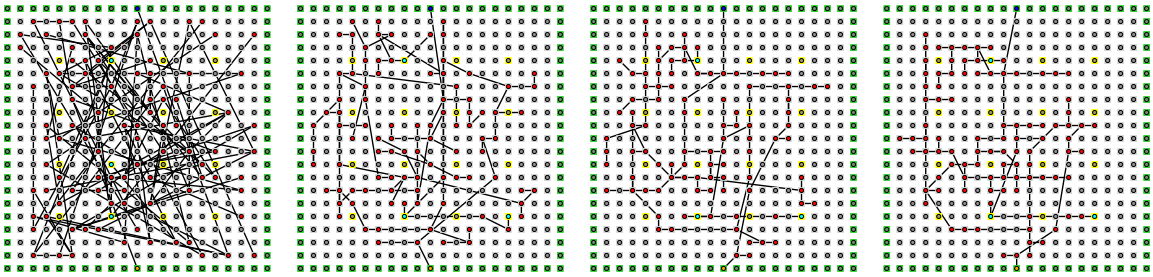
An experiment with similar configuration can be found in Figure 7.6, but we know compare the performance varying the dimension of the chosen subproblems k. Here, the number of unbound indices is fixed to $k_u = 30$. We observe similar behavior as for Figure 7.5 but choosing the worst subindices especially falls back in performance to choosing random indices for small k and large m.

Fixing $k = 100$ and $k_u = 50$, we plot intermediate placement results of Algorithm 8 after 1, 10 and 50 iterations in Figures 7.7 and 7.8. We fix the locations of the IO cells before the actual placement and use a random initialization. We can see that the initial placement is very bad, in the sense that the connecting edges are spread over the whole chip grid and cross each other, leading to a large QAP cost. With an increasing number of iterations, the placement gets a more grid-like structure with less crossings, leading to very preferable results for a potential subsequent routing. In Figure 7.8, the intermediate placements for a random instance with $m = 200$ and fixed IO locations is shown. We can see, that it takes more iterations to achieve a good placement on the first sight, than for $m = 100$.

7.5.2 CRC Example

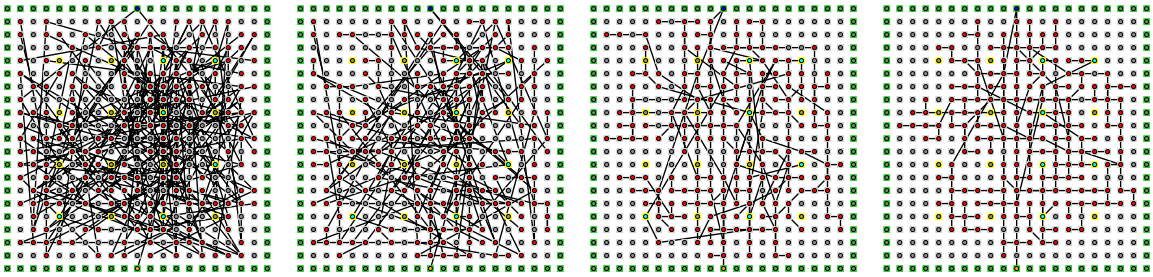
As a real-world example, we consider a simple 32-bit Cyclic Redundancy Check (CRC-32) algorithm with 8-bit parallel input. This is synthesized by the open-source tool Yosys[3] for the Lattice MachXO2

[3] https://yosyshq.net/yosys (last accessed September 19, 2025)



(a) Initial, cost 2492. (b) 1 it., cost 576. (c) 10 it., cost 388. (d) 50 it., cost 328.

Figure 7.7: Intermediate placement results for an exemplary generic example with 100 facilities. The initial random placement (a) is indicated along with the result of applying our algorithm for 1 iteration (b), 10 iterations (c) and 50 iterations (d). The placement of the two IO facilities is fixed and the corresponding QAP costs are indicated. See Figure 7.4 for a legend.



(a) Initial, cost 4954. (b) 1 it., cost 2780. (c) 10 it., cost 1068. (d) 50 it., cost 768.

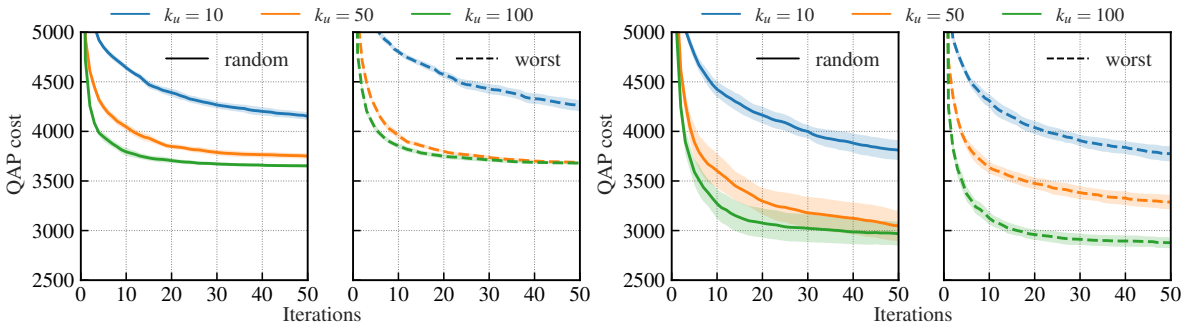
Figure 7.8: Intermediate placement results for an exemplary generic example with 200 facilities. The initial random placement (a) is indicated along with the result of applying our algorithm for 1 iteration (b), 10 iterations (c) and 50 iterations (d). The placement of the two IO facilities is fixed and the corresponding QAP costs are indicated. See Figure 7.4 for a legend.

architecture. Wide LUTs and CCU2 carry chains are forbidden, so that the synthesized output only contains 78 LUT-4s and 32 registers. The Lattice MachXO2 is a current technology that is available in sizes starting from 256 LUTs[4], so it is comparable to our demo architecture.

For this real-world example, we conduct experiments with a similar configuration to the one used in Figure 7.5. In contrast to the previous experiments, we now do not vary our problem size m but compare setups with fixed IO cells with the setup of optimizing the placement of these cells along with the remaining functional blocks. In Figure 7.9, we fix $k = 100$ and vary $k_u \in \{10, 50, 100\}$. We observe that fixing the IO cells leads to faster convergence and thus a worse placement than with the possibility of also optimizing the IO placements. Although, for unfixed IO cells, the uncertainty in the outcomes is larger than in the fixed case. Furthermore, we can see that the relative performance of only choosing $k_u = 10$ unbound indices compared to $k_u = 50, 100$ is worse than in the generic case (cf. Figure 7.5). The number of needed unbound indices is thus heavily dependent on the underlying problem structure.

In Figure 7.10 we depict the effect of varying k when k_u is fixed to 30. We observe that for choosing random problems, changing the subproblem size does not have a very large effect on the QAP cost.

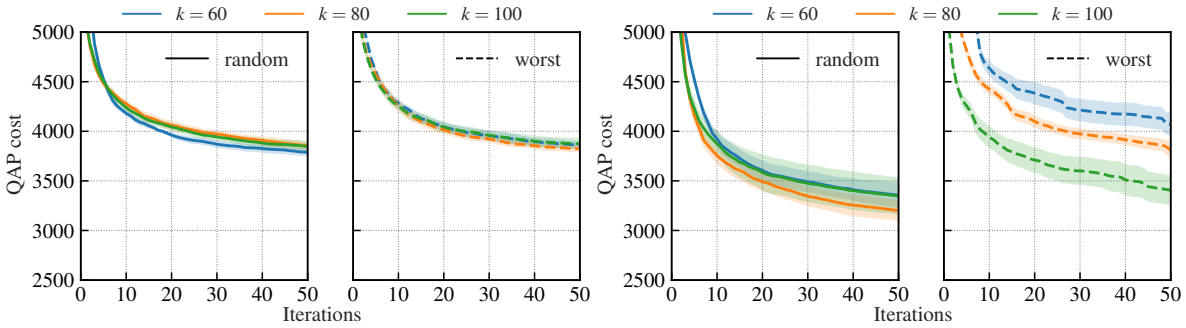
[4] https://www.latticesemi.com/view_document?document_id=38834 (last accessed September 19, 2025)



(a) Fixed IO cells.

(b) Unfixed IO cells.

Figure 7.9: Depicting the effect of varying k when k_u is fixed to a certain value, comparing choosing random subproblems in Line Algorithm 8 with choosing worst performing indices Equation (7.14). Here, $k_u = 30$ and the QAP cost is depicted over 50 iterations used in the CYCLICEXPANSION. We compare the costs for fixed IO cells (a) and unfixed IO cells (b) for the CRC-32.



(a) Fixed IO cells.

(b) Unfixed IO cells.

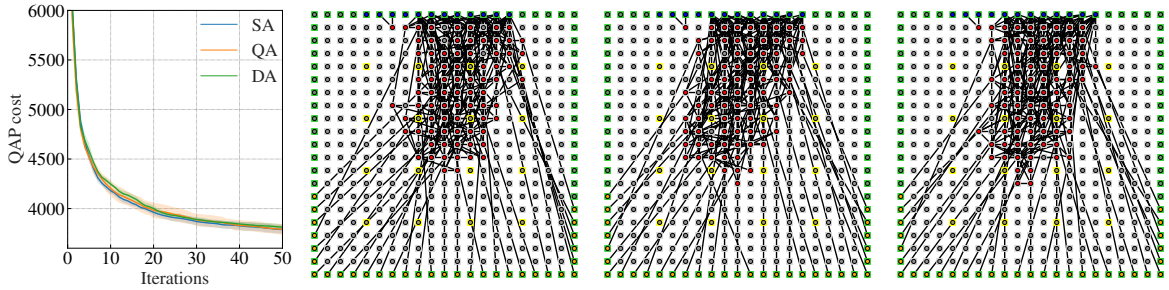
Figure 7.10: Depicting the effect of varying k when k is fixed to a certain value, comparing choosing random subproblems in Line Algorithm 8 with choosing worst performing indices Equation (7.14). Here, $k = 100$ and the QAP cost is depicted over 50 iterations used in the CYCLICEXPANSION. We compare the costs for fixed IO cells (a) and unfixed IO cells (b) for the CRC-32.

Thus, one already can achieve good placement results with a small problem size. However, choosing the subproblems greedily with Equation (7.14) is more sensitive in terms of performance outcomes for different problem sizes.

Again, similarly to the observations in Figures 7.7 and 7.8, we see that the placement also visually improves with an increasing number of iterations.

Lastly, we conduct experiments with QA and DA. We compare the performance of these two hardware solvers with SA on the CRC-32 example with fixed IO cells in Figure 7.11. We fix $k = 60$, $k_u = 30$ and choose the subproblems randomly. Figure 7.11(a) depicts the change of the QAP cost over an increasing number of iterations in the CYCLICEXPANSION. We find all solvers to perform equally well for this problem.

Figures 7.11(b) to 7.11(d) depict the placement results for specific runs after 50 iterations using QA, DA and SA, respectively. Since the QUBO problems in Equation (7.12) are well conditioned (integer valued and small dynamic ranges), real quantum hardware can achieve similar performance to classical



(a) QAP cost. (b) QA, cost 3732. (c) DA, cost 3790. (d) SA, cost 3758.

Figure 7.11: Performance comparison of the hardware solvers QA and DA with SA on the CRC-32 example. We choose random subproblems and fix $k = 60$ and $k = 30$. We depict the QAP cost over 50 iterations for the CYCLICEXPANSION (a) and exemplary placements after 50 iterations with QA (b), DA (c) and SA (d).

Table 7.1: QAP cost comparison of our CYCLICEXPANSION method to a *random* placement, as well as simulated annealing (SA) and an analytical placement (HeAP) as implemented in *nextpnr*. We use the CRC-32 example and synthesize for the generic architecture. Numbers are average QAP costs over 10 repetitions of the respective method. Lower is better.

Random	SA	HeAP	Ours (1 it)	Ours (2 it)	Ours (10 it)
11417.0	7017.6	7047.8	8192.8	6560.2	4762.0

solvers. This is an interesting result, since today's quantum technology is still in its infancy with limited computational power (number of qubits) and large proneness to errors. A detailed discussion on the effect of the conditioning of QUBO problems can be found in [45].

We also compare our CYCLICEXPANSION algorithm to two state-of-the-art placers implemented in the open-source framework *nextpnr* [258]. Table 7.1 depicts the performance comparison in terms of QAP cost of a simulated annealing (SA) placer [241], an analytical placer (HeAP) [40] and our algorithm. We can see that our method already outperforms the other placers after two iterations in terms of QAP cost. These results are very promising for future benchmarking of our algorithm on real FPGA hardware.

7.6 Conclusion

Our proposed method addresses the limitations of NISQ devices in solving large-scale QUBO problems by focusing on problem reformulation rather than subproblem decomposition. We introduce efficient encoding schemes that reduce problem size while maintaining solution quality. This reformulation not only increases sparsity in the QUBO matrix but also avoids the complexity introduced by additional constraints. Specifically, we tackle the QAP, which is notoriously hard to be solved with classical algorithms. With the notion of subpermutations, we find a new QUBO formulation for the unbalanced QAP without introducing dummy facilities, leading to a lower dimensionality. The problem of incorporating constraints for the set of allowed subpermutations is overcome by considering the iterative CYCLICEXPANSION algorithm. It is inspired by classical α-expansion, where optimization is performed over cyclic permutations instead of the entire solution space. This approach ensures scalability, allowing

for arbitrary problem sizes while remaining NISQ-aware. Moreover, initial solutions can easily be incorporated into this algorithm.

To demonstrate the effectiveness of our method, we apply it to the FPGA-placement problem, a critical task in chip design that significantly influences circuit performance, power consumption, and resource utilization. Our cyclic expansion algorithm refines FPGA-placements iteratively by selecting and optimizing cyclic permutations, seamlessly integrating hardware constraints into the selection process. This adaptability enables optimization within the constraints of NISQ hardware. Given the increasing complexity of FPGA architectures, traditional algorithms struggle to efficiently map functional blocks to chip grids. For comparison with classical methods, we integrate our algorithm into the open-source FPGA implementation software framework *nextpnr*. Experimental results on digital annealing devices and quantum annealers confirm the viability of our approach.

In this work, we consider binary flow matrices and Manhattan distances between locations on the FPGA chip. Real architectures can easily be integrated into our framework, by adapting the distance matrix and flow matrix correspondingly. Furthermore, our experiments are conducted on randomly generated problems as well as a small real-world circuit (CRC-32). We defer the investigation of large-scale use-cases as well as additional performance metrics such as the maximum clock rate of the final chip design to follow-up work, since we were more interested in theoretical properties.

CHAPTER 8

Conclusion

In recent years, ML has made remarkable advancements, enabling breakthroughs in various fields including computer vision, natural language processing, and scientific discovery. However, as datasets grow in scale and complexity, the underlying data properties—such as high dimensionality, potential outliers, and noise—pose significant challenges for traditional optimization methods. This limits the efficiency and scalability of ML models in the absence of extensive computing resources. QC offers a fundamentally new approach to tackle such problems by leveraging quantum mechanical effects. QO, in particular, holds the promise of efficiently solving CO tasks that are intractable for classical algorithms. However, the limitations of current NISQ devices, such as limited qubit availability and noise-induced errors, restrict the practical application of QO to real-world problems.

In this thesis, we investigated the relationship between the underlying data of a problem setting and the corresponding QO performance on NISQ devices. The primary goal was to address the computational challenges posed by NISQ hardware: the limited qubit availability and proneness to errors. By systematically analyzing the effects of data complexity on QO solvability and proposing strategies to mitigate these challenges, we developed an efficient framework for preprocessing QUBO problems. Regarding the limited scale of NISQ devices, we presented efficient, iterative algorithms for reducing the problem size. We outline the specific steps in the next section and conclude with a discussion on the broader impact of our work, along with potential future research directions.

8.1 Summary

In the first part of the thesis, we explored how data complexity impacts QUBO solvability, with a focus on SG analysis as a key metric for AQC performance in Chapter 3. For this, we examined QUBO embeddings of different ML tasks with a combinatorial nature. Through a detailed study of the corresponding QUBOs, we demonstrated that data complexity characteristics like separability and compactness directly influence the convergence behavior of QO algorithms. Interestingly, the results did not directly align with our intuition that a harder underlying problem also leads to worse solvability on quantum hardware. Furthermore, we proposed a generally applicable noise mitigation technique in Chapter 4 based on an MDP framework. Identifying the DR as a suitable precision measure, we found that it is more general than other measures used in the literature. Moreover, we introduced a principled B&B algorithm that iteratively adapts QUBO weights, while preserving an optimal solution. This not only enhanced the robustness of QUBO instances against quantum hardware errors but also

led to optimized resource usage for classical hardware solvers. Our approach significantly improved the reliability and accuracy of finding optimal solutions on current NISQ devices and outperforms state-of-the-art methods for precision reduction.

The second part of the thesis focused on overcoming the problem size limitations imposed by NISQ hardware. We introduced an iterative top-down D&C algorithm for QUBO decomposition in Chapter 5, allowing the efficient handling of large-scale CO problems. Special care was taken in the recombination of solutions, respecting the underlying structure of given constraints. This led to a trade-off between the amount of global correlation and local parallelizability of small-scale problems. In contrast to this top-down approach, a bottom-up variable and constraint generation method was developed in Chapter 6. It is inspired by Column Generation in ILP, which aims to expand QUBO formulations dynamically while preserving solution quality. Using LR, we proved an optimality criterion telling us when our generated variables contain an optimal solution, leading to a potentially large reduction of the problem size. This criterion relies on efficiently computable bounds: a lower bound can be obtained by optimizing the LR with Linear Programming and an upper bound is given by solving hardware-aware QUBO formuluations with NISQ hardware. The decision of which variable to add next is made by solving a pricing problem, which can also be done efficiently for specific problem structures. Apart from considering subproblems, we proposed different QUBO encoding strategies of the underlying problem in Chapter 7. By considering permutation-based optimization problems, we first presented a logarithmic encoding scheme to circumvent the quadratically large size of permutation matrices. Then, a cyclic expansion technique was presented, which natively integrates design constraints into the choice of the decision variables. It iteratively considers sets of disjoint cyclic permutations and solving a QUBO indicates which of these cycles should be applied to a given permutation. Obtaining a trade-off between problem size and number of iterations for convergence, the QUBO dimensionality can be conveniently adapted to the quantum hardware size.

The proposed methods were validated with regard to different ML tasks and real-world use-cases. Specifically we considered SVMs, clustering and VQ and conducted experiments with varying problem hardness. For the latter, we stumbled across the intriguing insight that a k-medoids-based formulation is more general than a KDE approach. For investigating the performance on real-world use-cases, we considered problems whose large dimensionality stems from the underlying data. We first examined the 3D scene reconstruction computer vision task of BA. For experimental evaluation, we considered high-resolution satellite images with millions of pixels and applied our D&C algorithm for introducing global correlations in extracted keypoints. We further presented a QUBO formulation for matching keypoints between images, resulting in high-quality image alignment. Integrating quantum kernels into our QUBO formulations, we obtained a combination of AQC and Quantum Gate Computing (QGC) by projecting the image data into an exponentially large Hilbert space. To demonstrate the effectiveness of our variable and constraint generation method, we used MAPF as an application. The goal is to find collision free paths between multiple agents simultaneously, leading to potentially intractable problem sizes for classical methods in finding an optimal solution. For obtaining upper bounds on the optimum needed in our optimality criterion, we presented different hardware-aware QUBO formulations indicating advantages over methods used in the literature. Comparing our algorithm to state-of-the-art methods, we showed potential benefits for upcoming quantum hardware. As a last real-world large-scale CO task, we considered the FPGA-placement, which is vital for the whole chip design process. Due to increasing chip sizes consisting of millions of functional blocks, classical methods often need unreasonable amounts of time and lead to unsatisfactory results. For all use-cases we conducted experiments on QA and DA devices, proving the viability of our methods.

8.2 Outlook

QO on NISQ devices is a rapidly evolving research field, with novel approaches emerging to address the limitations of current quantum hardware. With our proposed QUBO-based methods, we have developed effective techniques that are sensitive to data complexity by mitigating noise-induced errors while also respecting limited scale by using problem size reductions. However, many open questions remain and our methods for improving solvability and scalability on NISQ devices would greatly benefit from further research.

Regarding the relation of QO solvability and the complexity of the underlying data of an ML task, we mainly explored empirical approaches. Theoretical investigation would thus be very intriguing, especially with respect to the SG of the problem Hamiltonian. Even though exactly determining the SG can be notoriously hard, we have seen in Chapter 3 that bounds can be computed for estimation. Using such bounds in combination with a similar approach to our developed precision reduction method, it would be interesting to also develop a QUBO preprocessing method for enlarging the SG of the corresponding Hamiltonian. With a theoretical understanding of the effect of data complexity on the SG, one could device effective heuristics for an MDP, which could then be used by using PR [93, 94]. Apart from a theoretical investigation, efficient heuristics could also be learned within a RL [87] framework. Due to the continuous state and discrete action space, deep Q-learning [88] would be a good candidate for this investigation. This would not only provide us with insights into enlarging the SG but could also improve our noise mitigation technique.

Future work for problem size reduction building upon our developed methods could include a closer examination of the given constraints. Even though we adapted our D&C towards k-hot constraints, different problem structures could be integrated into the recombination procedure. Concerning variable generation, a valid initial solution could be designed more carefully in a way that respects the underlying constraints, similar to the method of Farkas pricing [35] in classical Column Generation. For improving Cyclic Expansion, a more informative approach on how to choose the cycles in setting up the QUBO formulation could be chosen, leveraging knowledge on the constraints. Theoretical insights on the convergence could be further integrated, giving us optimality guarantees. Lastly, we did not directly take the underlying hardware topology of NISQ devices into account, but limited our investigation towards reducing the problem size. Developing methods that take such graph topology constraints into account could also significantly improve QO performance. For example, embedding arbitrary graph structures into the underlying qubit topology of the D-Wave systems is achieved by chaining multiple physical qubits together into a single logical super-qubit. Reducing such chaining would not only lead to larger embeddable problems but also reduce hardware-induced noise.

In conclusion, this thesis provided theoretically sound and efficient algorithms for noise mitigation through QUBO preprocessing and dimensionality reduction. The effectiveness of our approaches has been shown through application to different ML tasks and large-scale CO use-cases. With the abstraction of the theoretical algorithmic concept and the specific use-case, our methods can also be conveniently adapted to different application areas. By bridging the gap between data complexity/scale and NISQ hardware limitations, this thesis provides a solid foundation for future advancements in QO. Moreover, the insights gained from this work are transferable to post-NISQ quantum computers, paving the way for scalable and robust quantum algorithms in the era of fault-tolerant quantum hardware.

Bibliography

[1] A. Sbihi and R. W. Eglese, *Combinatorial optimization and green logistics*, Annals of Operations Research **175** (2010) 159.

[2] G. Cornuejols, J. Peña and R. Tütüncü, *Optimization methods in finance*, Cambridge University Press, 2018.

[3] A. Ahmadi-Javid, P. Seyedi and S. S. Syam, *A survey of healthcare facility location*, Computers & Operations Research **79** (2017) 223.

[4] S. A. Gabriel, A. J. Conejo, J. D. Fuller, B. F. Hobbs and C. Ruiz, *Complementarity modeling in energy markets*, Springer Science & Business Media, 2012.

[5] F. Neukart et al., *Traffic flow optimization using a quantum annealer*, Frontiers in ICT **4** (2017) 29.

[6] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams and N. De Freitas, *Taking the human out of the loop: A review of Bayesian optimization*, Proceedings of the IEEE **104** (2015) 148.

[7] F. Glover and M. Laguna, *Tabu search*, Springer, 1998.

[8] D. E. Goldberg and C. H. Kuo, *Genetic algorithms in pipeline optimization*, Journal of Computing in Civil Engineering **1** (1987) 128.

[9] A. P. Punnen, *The quadratic unconstrained binary optimization problem*, Springer, 2022.

[10] D. Rehfeldt, T. Koch and Y. Shinano, *Faster exact solution of sparse MaxCut and QUBO problems*, Mathematical Programming Computation **15** (2023) 445.

[11] E. Alpaydin, *Machine learning*, MIT press, 2021.

[12] Y. LeCun, Y. Bengio and G. Hinton, *Deep learning*, Nature **521** (2015) 436.

[13] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman and G. Hinton, *Backpropagation and the brain*, Nature Reviews Neuroscience **21** (2020) 335.

[14] D. Silver et al., *Mastering the game of Go with deep neural networks and tree search*, Nature **529** (2016) 484.

[15] J. Jumper et al., *Highly accurate protein structure prediction with AlphaFold*, Nature **596** (2021) 583.

[16] J. Achiam et al., *GPT-4 technical report*, arXiv preprint arXiv:2303.08774 (2023).

[17] R. Rombach, A. Blattmann, D. Lorenz, P. Esser and B. Ommer, "High-resolution image synthesis with latent diffusion models", *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2022 10684.

[18] J. Ho et al., "Video diffusion models", *Advances in Neural Information Processing Systems (NeurIPS) 36*, 2022 8633.

[19] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*, Cambridge University Press, 2010.

[20] H. Buhrman, R. Cleve and W. Van Dam, *Quantum entanglement and communication complexity*, SIAM Journal on Computing **30** (2001) 1829.

[21] Y. S. Weinstein, M. Pravia, E. Fortunato, S. Lloyd and D. G. Cory, *Implementation of the quantum Fourier transform*, Physical Review Letters **86** (2001) 1889.

[22] P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM review **41** (1999) 303.

[23] L. K. Grover, "A fast quantum mechanical algorithm for database search", *ACM Symposium on Theory of Computing (STOC)*, ACM, 1996 212.

[24] Y. Cao, J. Romero and A. Aspuru-Guzik, *Potential of quantum computing for drug discovery*, IBM Journal of Research and Development **62** (2018) 1.

[25] B. Bauer, S. Bravyi, M. Motta and G. K.-L. Chan, *Quantum algorithms for quantum chemistry and quantum materials science*, Chemical Reviews **120** (2020) 12685.

[26] J. Biamonte et al., *Quantum machine learning*, Nature **549** (2017) 195.

[27] A. Abbas et al., *Challenges and opportunities in quantum optimization*, Nature Reviews Physics **6** (2024) 1.

[28] Z. Bian, F. Chudak, W. Macready and G. Rose, *The Ising model: teaching an old problem new tricks*, tech. rep., D-Wave Systems, 2010.

[29] J. Preskill, *Quantum computing in the NISQ era and beyond*, Quantum **2** (2018) 79.

[30] C. Cortes, *Support-vector networks*, Machine Learning (1995).

[31] A. Saxena et al., *A review of clustering techniques and developments*, Neurocomputing **267** (2017) 664.

[32] R. Gray, *Vector quantization*, IEEE ASSP Magazine **1** (1984) 4.

[33] T. Albash and D. A. Lidar, *Adiabatic quantum computation*, Reviews of Modern Physics **90** (2018) 015002.

[34] B. Triggs, P. F. McLauchlan, R. I. Hartley and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis", *Proceedings of the International Workshop on Vision Algorithms*, Springer, 1999 298.

[35] M. E. Lübbecke, *Column generation*, Encyclopedia of Operations Research and Management Science **17** (2010) 18.

[36] L. A. Wolsey, *Integer programming*, John Wiley & Sons, 2020.

[37] R. Stern et al., "Multi-agent pathfinding: Definitions, variants, and benchmarks", *Proceedings of the 12th International Symposium on Combinatorial Search (SoCS)*, AAAI Press, 2019 151.

[38] E. L. Lawler, *The quadratic assignment problem*, Management Science **9** (1963) 586.

[39] Y. Boykov, O. Veksler and R. Zabih, *Fast approximate energy minimization via graph cuts*, IEEE Transactions on Pattern Analysis and Machine Intelligence **23** (2001) 1222.

[40] M. Gort and J. H. Anderson, "Analytical placement for heterogeneous FPGAs", *Proceedings of the 22nd International Conference on Field Programmable Logic and Applications (FPL)*, IEEE, 2012 143.

[41] N. Piatkowski et al., "Towards bundle adjustment for satellite imaging via quantum machine learning", *Proceedings of the 25th International Conference on Information Fusion (FUSION)*, IEEE, 2022 1, DOI: https://doi.org/10.23919/FUSION49751.2022.9841388.

[42] T. Gerlach et al., "FPGA-placement via quantum annealing", *Proceedings of the 32nd ACM/SIGDA International Symposium on Field Programmable Gate Arrays (ISFPGA)*, ACM, 2024 43, DOI: https://doi.org/10.1145/3626202.3637619.

[43] T. Gerlach and S. Mücke, "Investigating the relation between problem hardness and QUBO properties", *Proceedings of the 22nd International Symposium on Intelligent Data Analysis (IDA)*, Springer, 2024 171, DOI: https://doi.org/10.1007/978-3-031-58553-1_14.

[44] T. Gerlach et al., "Quantum optimization for FPGA-placement", *Proceedings of the 2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, IEEE, 2024 637, DOI: https://doi.org/10.1109/QCE60285.2024.00080.

[45] S. Mücke, T. Gerlach and N. Piatkowski, *Optimum-preserving QUBO parameter compression*, Quantum Machine Intelligence **7** (2025) 1, DOI: https://doi.org/10.1007/s42484-024-00219-3.

[46] T. Gerlach, L. K. Lee, F. Barbaresco and N. Piatkowski, "Hybrid quantum-classical multi-agent pathfinding", *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, To appear, PMLR, 2025, DOI: https://doi.org/10.48550/arXiv.2501.14568.

[47] T. Gerlach and N. Piatkowski, "Dynamic range reduction via branch-and-bound", *Proceedings of the 2025 IEEE International Conference on Quantum Computing and Engineering (QCE)*, To appear, IEEE, 2025, DOI: https://doi.org/10.48550/arXiv.2409.10863.

[48] T. Gerlach, S. Mücke and C. Bauckhage, "Kernel k-Medoids as General Vector Quantization", *Proceedings of the 2025 IEEE International Conference on Quantum Artificial Intelligence (QAI)*, To appear, IEEE, 2025, DOI: https://doi.org/10.48550/arXiv.2506.04786.

[49] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge university press, 2004.

[50] B. H. Korte, J. Vygen, B. Korte and J. Vygen, *Combinatorial optimization*, Springer, 2011.

[51] M. Eskandarpour, P. Dejax, J. Miemczyk and O. Péton,
Sustainable supply chain network design: An optimization-oriented review,
Omega **54** (2015) 11.

[52] M. L. Pinedo, *Scheduling*, Springer, 2012.

[53] S. Held, B. Korte, D. Rautenbach and J. Vygen, *Combinatorial optimization in VLSI design*,
Combinatorial Optimization (2011) 33.

[54] G. Ausiello et al., *Complexity and approximation: Combinatorial optimization problems and their approximability properties*, Springer Science & Business Media, 2012.

[55] A. Lucas, *Ising formulations of many NP problems*, Frontiers in physics **2** (2014) 5.

[56] G. Kochenberger, F. Glover, B. Alidaee and K. Lewis,
Using the unconstrained quadratic program to model and solve Max 2-SAT problems,
International Journal of Operational Research **1** (2005) 89.

[57] T. Stollenwerk, E. Lobe and M. Jung, "Flight gate assignment with a quantum annealer",
International Workshop on Quantum Technology and Optimization Problems, Springer, 2019 99.

[58] C. Bauckhage, N. Piatkowski, R. Sifa, D. Hecker and S. Wrobel,
"A QUBO formulation of the k-medoids problem",
Proceedings of the Conference on "Lernen, Wissen, Daten, Analysen" (LWDA),
CEUR-WS.org, 2019 54.

[59] C. Bauckhage, C. Ojeda, R. Sifa and S. Wrobel,
"Adiabatic quantum computing for kernel $k = 2$ means clustering",
Proceedings of the Conference on "Lernen, Wissen, Daten, Analysen" (LWDA),
CEUR-WS.org, 2018 21.

[60] S. Mücke, R. Heese, S. Müller, M. Wolter and N. Piatkowski,
Feature selection on quantum computers, Quantum Machine Intelligence **5** (2023) 11.

[61] P. M. Pardalos and S. Jha,
Complexity of uniqueness and local search in quadratic 0–1 programming,
Operations Research Letters **11** (1992) 119.

[62] S. Kirkpatrick, C. D. Gelatt Jr and M. P. Vecchi, *Optimization by simulated annealing*,
Science **220** (1983) 671.

[63] L. Gurobi Optimization, *Gurobi Optimizer Reference Manual*, 2025,
URL: https://www.gurobi.com.

[64] IBM Corporation, *IBM ILOG CPLEX Optimization Studio CPLEX User's Manual*, 2025, URL:
https://www.ibm.com/products/ilog-cplex-optimization-studio.

[65] G. Kochenberger et al., *The unconstrained binary quadratic programming problem: A survey*,
Journal of Combinatorial Optimization **28** (2014) 58.

[66] S. G. Brush, *History of the Lenz-Ising model*, Reviews of Modern Physics **39** (1967) 883.

[67] N. Mohseni, P. L. McMahon and T. Byrnes,
Ising machines as hardware solvers of combinatorial optimization problems,
Nature Reviews Physics **4** (2022) 363.

[68] W. A. Borders et al., *Integer factorization using stochastic magnetic tunnel junctions*, Nature **573** (2019) 390.

[69] T. Byrnes, K. Yan and Y. Yamamoto, *Accelerated optimization problem search using Bose–Einstein condensation*, New Journal of Physics **13** (2011) 113025.

[70] S. Matsubara et al., "Digital annealer for high-speed solving of combinatorial optimization problems and its applications", *Proceedings of the 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, IEEE, 2020 667.

[71] S. Mücke, N. Piatkowski and K. Morik, "Learning bit by bit: Extracting the essence of machine learning.", *Proceedings of the Conference on "Lernen, Wissen, Daten, Analysen" (LWDA)*, CEUR-WS.org, 2019 144.

[72] H. Kagawa et al., "Fully-pipelined architecture for simulated annealing-based QUBO solver on the FPGA", *Proceedings of the 8th International Symposium on Computing and Networking (CANDAR)*, IEEE, 2020 39.

[73] R. Yasudo et al., "Adaptive bulk search: Solving quadratic unconstrained binary optimization problems on multiple GPUs", *Proceedings of the 49th International Conference on Parallel Processing (ICPP)*, ACM, 2020 1.

[74] M. Tao et al., "A work-time optimal parallel exhaustive search algorithm for the QUBO and the Ising model, with GPU implementation", *Proceedings of the 2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, IEEE, 2020 557.

[75] Y. Yamamoto, T. Leleu, S. Ganguli and H. Mabuchi, *Coherent Ising machines—Quantum optics and neural network Perspectives*, Applied Physics Letters **117** (2020).

[76] S. H. G. Mastiyage Don, Y. Inui, S. Kako, Y. Yamamoto and T. Aonishi, *Mean-field coherent Ising machines with artificial Zeeman terms*, Journal of Applied Physics **134** (2023).

[77] V. Vapnik, *The nature of statistical learning theory*, Springer science & business media, 2013.

[78] J. Mercer, *Functions of positive and negative type, and their connection the theory of integral equations*, Philosophical Transactions of the Royal Society of London **209** (1909) 415.

[79] M. Schuld and F. Petruccione, *Supervised learning with quantum computers*, vol. 17, Springer, 2018.

[80] S. Lloyd, *Least squares quantization in PCM*, IEEE Transactions on Information Theory **28** (1982) 129.

[81] R. Jenssen, J. C. Principe, D. Erdogmus and T. Eltoft, *The Cauchy–Schwarz divergence and Parzen windowing: Connections to graph theory and Mercer kernels*, Journal of the Franklin Institute **343** (2006) 614.

[82] S. Kullback and R. A. Leibler, *On information and sufficiency*,
The Annals of Mathematical Statistics **22** (1951) 79.

[83] C. Bauckhage, R. Ramamurthy and R. Sifa, "Hopfield networks for vector quantization",
Proceedings of the 29th International Conference on Artificial Neural Networks (ICANN),
Springer, 2020 192.

[84] T. Lehn-Schiøler, A. Hegde, D. Erdogmus and J. C. Principe,
Vector quantization using information theoretic concepts, Natural Computing **4** (2005) 39.

[85] J.-W. Xu, A. R. Paiva, I. Park and J. C. Principe,
A reproducing kernel Hilbert space framework for information-theoretic learning,
IEEE Transactions on Signal Processing **56** (2008) 5891.

[86] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf and A. Smola,
A kernel two-sample test, The Journal of Machine Learning Research **13** (2012) 723.

[87] D. Bertsekas, *Reinforcement learning and optimal control*, vol. 1, Athena Scientific, 2019.

[88] C. J. Watkins and P. Dayan, *Q-learning*, Machine Learning **8** (1992) 279.

[89] R. S. Sutton, D. McAllester, S. Singh and Y. Mansour,
"Policy gradient methods for reinforcement learning with function approximation", 1999 1057.

[90] R. Bellman, *A Markovian decision process*,
Indiana University Mathematics Journal **6** (1957) 679.

[91] R. Bellman, *On the theory of dynamic programming*,
Proceedings of the National Academy of Sciences **38** (1952) 716.

[92] H. Van Hasselt, A. Guez and D. Silver, "Deep reinforcement learning with double q-learning",
Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI), 1,
AAAI Press, 2016.

[93] D. P. Bertsekas, J. N. Tsitsiklis and C. Wu, *Rollout algorithms for combinatorial optimization*,
Journal of Heuristics **3** (1997) 245.

[94] D. Bertsekas, *Rollout, policy iteration, and distributed reinforcement learning*,
Athena Scientific, 2021.

[95] V. Steinbiss, B.-H. Tran and H. Ney, "Improvements in beam search",
Proceedings of the 3rd International Conference on Spoken Language Processing (ICLSP),
ISCA, 1994 2143.

[96] A. D. King et al., *Computational supremacy in quantum simulation*,
arXiv preprint arXiv:2403.00910 (2024).

[97] A. D. King et al., *Beyond-classical computation in quantum simulation*,
Science (2025) eado6285.

[98] D. Coppersmith, *An approximate Fourier transform useful in quantum factoring*,
arXiv preprint quant-ph/0201067 (2002).

[99] V. Havlicek et al., *Supervised learning with quantum-enhanced feature spaces*,
Nature **567** (2019) 209.

[100] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster and J. I. Latorre, *Data re-uploading for a universal quantum classifier*, Quantum **4** (2020) 226.

[101] M. Born and V. Fock, *Beweis des adiabatensatzes*, Zeitschrift für Physik **51** (1928) 165.

[102] E. Farhi, J. Goldstone, S. Gutmann and M. Sipser, *Quantum computation by adiabatic evolution*, arXiv preprint quant-ph/0001106 (2000).

[103] J. Roland and N. J. Cerf, *Quantum search by local adiabatic evolution*, Physical Review A **65** (2002) 042308.

[104] D. Aharonov et al., *Adiabatic quantum computation is equivalent to standard quantum computation*, SIAM Review **50** (2008) 755.

[105] B. Altshuler, H. Krovi and J. Roland, *Adiabatic quantum optimization fails for random instances of NP-complete problems*, arXiv preprint arXiv:0908.2782 (2009).

[106] T. S. Cubitt, D. Perez-Garcia and M. M. Wolf, *Undecidability of the spectral gap*, Nature **528** (2015) 207.

[107] A. Peruzzo et al., *A variational eigenvalue solver on a photonic quantum processor*, Nature Communications **5** (2014) 4213.

[108] J. Tilly et al., *The variational quantum eigensolver: a review of methods and best practices*, Physics Reports **986** (2022) 1.

[109] Y. Cao et al., *Quantum chemistry in the age of quantum computing*, Chemical Reviews **119** (2019) 10856.

[110] A. Kandala et al., *Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets*, Nature **549** (2017) 242.

[111] E. Farhi, J. Goldstone and S. Gutmann, *A quantum approximate optimization algorithm*, arXiv preprint arXiv:1411.4028 (2014).

[112] S. Hadfield et al., *From the quantum approximate optimization algorithm to a quantum alternating operator ansatz*, Algorithms **12** (2019) 34.

[113] H. F. Trotter, *On the product of semi-groups of operators*, Proceedings of the American Mathematical Society **10** (1959) 545.

[114] M. J. Powell, *A direct search optimization method that models the objective and constraint functions by linear interpolation*, Springer, 1994.

[115] J. C. Spall, *An overview of the simultaneous perturbation method for efficient optimization*, Johns Hopkins APL Technical Digest **19** (1998) 482.

[116] B. Apolloni, C. Carvalho and D. De Falco, *Quantum stochastic optimization*, Stochastic Processes and their Applications **33** (1989) 233.

[117] B. Apolloni, N. Cesa-Bianchi and D. De Falco, "A numerical implementation of "quantum annealing"", *Proceedings of the 2nd International Conference on Stochastic Processes, Physics and Geometry*, World Scientific, 1990 97.

[118] T. Kadowaki and H. Nishimori, *Quantum annealing in the transverse Ising model*, Physical Review E **58** (1998) 5355.

[119] J. Brooke, D. Bitko, Rosenbaum and G. Aeppli, *Quantum annealing of a disordered magnet*, Science **284** (1999) 779.

[120] M. Johnson et al., *Quantum annealing with manufactured spins*, Nature **473** (2011) 194.

[121] T. Lanting et al., *Entanglement in a quantum annealing processor*, Physical Review X **4** (2014) 1.

[122] L. Henriet et al., *Quantum computing with neutral atoms*, Quantum **4** (2020) 327.

[123] D. Crawford, A. Levit, N. Ghadermarzy, J. S. Oberoi and P. Ronagh, *Reinforcement learning using quantum boltzmann machines*, Quantum Information & Computation **18** (2018) 51.

[124] A. Levit et al., *Free energy-based reinforcement learning using a quantum processor*, arXiv preprint arXiv:1706.00074 (2017).

[125] A. R. Calderbank, E. M. Rains, P. M. Shor and N. J. Sloane, *Quantum error correction via codes over GF(4)*, IEEE Transactions on Information Theory **44** (1998) 1369.

[126] W. H. Zurek, *Decoherence, einselection, and the quantum origins of the classical*, Reviews of Modern Physics **75** (2003) 715.

[127] K. Temme, S. Bravyi and J. M. Gambetta, *Error mitigation for short-depth quantum circuits*, Physical Review Letters **119** (2017) 180509.

[128] L. Viola and S. Lloyd, *Dynamical suppression of decoherence in two-state quantum systems*, Physical Review A **58** (1998) 2733.

[129] B. Fauseweh, *Quantum many-body simulations on digital quantum computers: State-of-the-art and future challenges*, Nature Communications **15** (2024) 2123.

[130] D. Peral-García, J. Cruz-Benito and F. J. García-Peñalvo, *Systematic literature review: Quantum machine learning and its applications*, Computer Science Review **51** (2024) 100619.

[131] M. Kjaergaard et al., *Superconducting qubits: Current state of play*, Annual Review of Condensed Matter Physics **11** (2020) 369.

[132] C. D. Bruzewicz, J. Chiaverini, R. McConnell and J. M. Sage, *Trapped-ion quantum computing: Progress and challenges*, Applied Physics Reviews **6** (2019).

[133] S. Slussarenko and G. J. Pryde, *Photonic quantum information processing: A concise review*, Applied Physics Reviews **6** (2019).

[134] M. A. Quantum et al., *Interferometric single-shot parity measurement in InAs–Al hybrid devices*, Nature **638** (2025) 651.

[135] V. Guruswami and P. Raghavendra, *Hardness of learning halfspaces with noise*, SIAM Journal on Computing **39** (2009) 742.

[136] S. Dooley, G. Kells, H. Katsura and T. C. Dorlas, *Simulating quantum circuits by adiabatic computation: Improved spectral gap bounds*, Physical Review A **101** (2020) 042302.

[137] M. H. Amin and V. Choi,
First-order quantum phase transition in adiabatic quantum computation,
Physical Review A **80** (2009) 062326.

[138] M. Werner, A. García-Sáez and M. P. Estarellas,
Bounding first-order quantum phase transitions in adiabatic quantum computing,
Physical Review Research **5** (2023) 043236.

[139] V. Choi, *The effects of the problem Hamiltonian parameters on the minimum spectral gap in adiabatic quantum optimization*, Quantum Information Processing **19** (2020) 90.

[140] A. Braida and S. Martiel, *Anti-crossings and spectral gap during quantum adiabatic evolution*, Quantum Information Processing **20** (2021) 260.

[141] K. S. Rai, J.-F. Chen, P. Emonts and J. Tura,
Spectral gap optimization for enhanced adiabatic state preparation,
arXiv preprint arXiv:2409.15433 (2024).

[142] S. Nagies et al., *Boosting quantum annealing performance through direct polynomial unconstrained binary optimization*, arXiv preprint arXiv:2412.04398 (2024).

[143] C. Roch, A. Impertro and C. Linnhoff-Popien,
"Cross entropy optimization of constrained problem Hamiltonians for quantum annealing",
International Conference on Computational Science (ICCS), Springer, 2021 60.

[144] C. Roch, D. Ratke, J. Nüßlein, T. Gabor and S. Feld, *The effect of penalty factors of constrained Hamiltonians on the eigenspectrum in quantum annealing*,
ACM Transactions on Quantum Computing **4** (2023) 1.

[145] M. S. Benkner, V. Golyanik, C. Theobalt and M. Moeller,
"Adiabatic quantum graph matching with permutation matrix constraints",
Proceedings of the 8th International Conference on 3D Vision (3DV), IEEE, 2020 583.

[146] E. Alessandroni, S. Ramos-Calderer, I. Roth, E. Traversi and L. Aolita,
Alleviating the quantum Big-M problem, arXiv preprint arXiv:2307.10379 (2023).

[147] D. Willsch, M. Willsch, H. De Raedt and K. Michielsen,
Support vector machines on the D-Wave quantum annealer,
Computer Physics Communications **248** (2020) 107006.

[148] P. Date, D. Arthur and L. ¡Pusey-Nazzaro,
QUBO formulations for training machine learning models, Scientific Reports **11** (2021) 10029.

[149] D. Aloise, A. Deshpande, P. Hansen and P. Popat,
NP-hardness of Euclidean sum-of-squares clustering, Machine Learning **75** (2009) 245.

[150] R. Fisher, *On the "probable error" of a coefficient of correlation deduced from a small sample.*,
Metron **1** (1921) 3.

[151] J. E. Dennis Jr and R. E. Welsch, *Techniques for nonlinear least squares and robust regression*,
Communications in Statistics-Simulation and Computation **7** (1978) 345.

[152] J. N. Franklin, *Matrix theory*, Courier Corporation, 2012.

[153] Y. Choukroun, E. Kravchik, F. Yang and P. Kisilev,
"Low-bit quantization of neural networks for efficient inference", *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, IEEE, 2019 3009.

[154] S. Ermon, C. Gomes, A. Sabharwal and B. Selman,
"Taming the curse of dimensionality: Discrete integration by hashing and optimization", *Proceedings of the 30th International Conference on Machine Learning (ICML)*, PMLR, 2013 334.

[155] G. Brown, "A new perspective for information theoretic feature selection", *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, PMLR, 2009 49.

[156] T. F. Rønnow et al., *Defining and detecting quantum speedup*, Science **345** (2014) 420.

[157] D. Oku, M. Tawada, S. Tanaka and N. Togawa,
How to reduce the bit-width of an Ising model by adding auxiliary spins,
IEEE Transactions on Computers **71** (2020) 223.

[158] O. Şeker, N. Tanoumand and M. Bodur, *Digital annealer for quadratic unconstrained binary optimization: A comparative performance analysis*,
Applied Soft Computing **127** (2022) 109367.

[159] M. Booth, S. P. Reinhardt and A. Roy,
Partitioning optimization problems for hybrid classical/quantum execution, tech. rep.,
D-Wave Systems Inc., 2017, URL: https://docs.ocean.dwavesys.com/projects/qbsolv/en/latest/_downloads/bd15a2d8f32e587e9e5997ce9d5512cc/qbsolv_techReport.pdf.

[160] T. Vyskočil, S. Pakin and H. N. Djidjev,
"Embedding inequality constraints for quantum annealing optimization", *Proceedings of the First International Workshop on Quantum Technology and Optimization Problems (QTOP)*, ed. by S. Feld and C. Linnhoff-Popien, vol. 1, Springer, 2019 11.

[161] Y. Yachi, Y. Mukasa, M. Tawada and N. Togawa,
"Efficient coefficient bit-width reduction method for ising machines",
Proceedings of the 40th International Conference on Consumer Electronics (ICCE), IEEE, 2022 1.

[162] G. J. Mooney, S. U. Tonetto, C. D. Hill and L. C. Hollenberg,
Mapping NP-hard problems to restricted adiabatic quantum architectures,
arXiv preprint arXiv:1911.00249 (2019).

[163] Y. Yachi, M. Tawada and N. Togawa,
"A bit-width reducing method for Ising models guaranteeing the ground-state output",
Proceedings of the 36th International System-on-Chip Conference (SOCC), IEEE, 2023 1.

[164] T. Stollenwerk et al.,
Quantum annealing applied to de-conflicting optimal trajectories for air traffic management,
IEEE Transactions on Intelligent Transportation Systems **21** (2019) 285.

[165] A. Verma and M. Lewis,
Penalty and partitioning techniques to improve performance of QUBO solvers,
Discrete Optimization **44** (2022) 100594.

[166] B. Seeber, *Handbook of Applied Superconductivity, Volume 2*, vol. 2, CRC press, 1998.

[167] G. Ballou, *Handbook for sound engineers*, Taylor & Francis, 2013.

[168] E. Boros, P. L. Hammer and G. Tavares,
Preprocessing of unconstrained quadratic binary optimization, tech. rep.,
Technical Report RRR 10-2006, RUTCOR, 2006.

[169] M. Lewis and F. Glover, *Quadratic unconstrained binary optimization problem preprocessing: Theory and empirical analysis*, Networks **70** (2017) 79.

[170] F. Glover, M. Lewis and G. Kochenberger, *Logical and inequality implications for reducing the size and difficulty of quadratic unconstrained binary optimization problems*,
European Journal of Operational Research **265** (2018) 829.

[171] E. Boros, P. L. Hammer, R. Sun and G. Tavares, *A max-flow approach to improved lower bounds for quadratic unconstrained binary optimization (QUBO)*,
Discrete Optimization **5** (2008) 501.

[172] P. L. Hammer, P. Hansen and B. Simeone,
Roof duality, complementation and persistency in quadratic 0–1 optimization,
Mathematical programming **28** (1984) 121.

[173] D. Biesner, T. Gerlach, C. Bauckhage, B. Kliem and R. Sifa,
"Solving subset sum problems using quantum inspired optimization algorithms with applications in auditing and financial data analysis", *Proceedings of the 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, 2022 903.

[174] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd,
New York, NY, USA: Cambridge University Press, 2003.

[175] G. Rosenberg, M. Vazifeh, B. Woods and E. Haber,
Building an iterative heuristic solver for a quantum annealer,
Computational Optimization and Applications **65** (2016) 845.

[176] G. Bass, M. Henderson, J. Heath and J. Dulny III,
Optimizing the optimizer: decomposition techniques for quantum annealing,
Quantum Machine Intelligence **3** (2021) 10.

[177] S. Fortunato, *Community detection in graphs*, Physics Reports **486** (2010) 75.

[178] S. E. Schaeffer, *Graph clustering*, Computer Science Review **1** (2007) 27.

[179] S. Bravyi, A. Kliesch, R. Koenig and E. Tang,
Obstacles to variational quantum optimization from symmetry protection,
Physical review letters **125** (2020) 260505.

[180] D. J. Egger, J. Mareček and S. Woerner, *Warm-starting quantum optimization*,
Quantum **5** (2021) 479.

[181] G. G. Guerreschi, *Solving quadratic unconstrained binary optimization with divide-and-conquer and quantum algorithms*, arXiv preprint arXiv:2101.07813 (2021).

[182] J. Li, M. Alam and S. Ghosh,
Large-scale quantum approximate optimization via divide-and-conquer, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **42** (2022) 1852.

[183] Y. Ruan, X. Xue and Y. Shen, *Quantum image processing: opportunities and challenges*, Mathematical Problems in Engineering **2021** (2021) 6671613.

[184] Z. Wang, M. Xu and Y. Zhang, *Review of quantum image processing*, Archives of Computational Methods in Engineering **29** (2022) 737.

[185] M. E. Haque, M. Paul, A. Ulhaq and T. Debnath,
Advanced quantum image representation and compression using a DCT-EFRQI approach, Scientific Reports **13** (2023) 4129.

[186] S. K. Deb and W. D. Pan,
Quantum image compression: Fundamentals, algorithms, and advances,
Computers **13** (2024) 185.

[187] J. Mu, X. Li, X. Zhang and P. Wang,
Quantum implementation of the classical guided image filtering algorithm,
Scientific Reports **15** (2025) 493.

[188] S. Caraiman and V. I. Manta, *Histogram-based segmentation of quantum images*, Theoretical Computer Science **529** (2014) 46.

[189] X.-W. Yao et al.,
Quantum image processing and its application to edge detection: theory and experiment, Physical Review X **7** (2017) 031041.

[190] A. Geng, A. Moghiseh, C. Redenbach and K. Schladitz,
A hybrid quantum image edge detector for the NISQ era,
Quantum Machine Intelligence **4** (2022) 15.

[191] W. Liu and L. Wang,
Quantum image edge detection based on eight-direction Sobel operator for NEQR,
Quantum Information Processing **21** (2022) 190.

[192] N.-R. Zhou, X.-X. Liu, Y.-L. Chen and N.-S. Du,
Quantum k-nearest-neighbor image classification algorithm based on KL transform,
International Journal of Theoretical Physics **60** (2021) 1209.

[193] S. Das, J. Zhang, S. Martina, D. Suter and F. Caruso,
Quantum pattern recognition on real quantum processing units,
Quantum Machine Intelligence **5** (2023) 16.

[194] Y. Zhang, K. Lu, K. Xu, Y. Gao and R. Wilson,
Local feature point extraction for quantum images,
Quantum Information Processing **14** (2015) 1573.

[195] S. Yuan, W. Lin, B. Hang and H. Meng, *Quantum fast corner detection algorithm*, Quantum Information Processing **22** (2023) 313.

[196] N. Jiang, Y. Dang and J. Wang, *Quantum image matching*,
Quantum Information Processing **15** (2016) 3543.

[197] M. S. Benkner et al., "Q-match: Iterative shape matching via quantum annealing", *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, 2021 7586.

[198] S. Mücke and T. Gerlach, "Efficient light source placement using quantum computing", *Proceedings of the Conference on "Lernen, Wissen, Daten, Analysen" (LWDA)*, CEUR-WS.org, 2023.

[199] D. De Santis, S. Tirone, S. Marmi and V. Giovannetti, *Optimized QUBO formulation methods for quantum computing*, arXiv preprint arXiv:2406.07681 (2024).

[200] J. A. Montañez-Barrera, D. Willsch, A. Maldonado-Romo and K. Michielsen, *Unbalanced penalization: A new approach to encode inequality constraints of combinatorial problems for quantum optimization algorithms*, Quantum Science and Technology **9** (2024) 025022.

[201] U. Von Luxburg, *A tutorial on spectral clustering*, Statistics and Computing **17** (2007) 395.

[202] T. Häner, K. E. Booth, S. E. Borujeni and E. Y. Zhu, *Solving QUBOs with a quantum-amenable branch and bound method*, arXiv preprint arXiv:2407.20185 (2024).

[203] Y. I. Abdel-Aziz, H. Karara and M. Hauck, *Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry*, Photogrammetric Engineering & Remote Sensing **81** (2015) 103.

[204] R. I. Hartley, *In defense of the eight-point algorithm*, IEEE Transactions on Pattern Analysis and Machine Intelligence **19** (1997) 580.

[205] G. D. Evangelidis and C. Bauckhage, *Efficient subframe video alignment using short descriptors*, IEEE Transactions on Pattern Analysis and Machine Intelligence **35** (2013) 2371.

[206] T. Presles, C. Enderli, R. Bricout, F. Aligne and F. Barbaresco, "Phase-coded radar waveform AI-based augmented engineering and optimal design by Quantum Annealing", preprint, 2021.

[207] D. G. Lowe, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision **60** (2004) 91.

[208] P. F. Alcantarilla, A. Bartoli and A. J. Davison, "KAZE features", *Proceedings of the 12th European Conference on Computer Vision (ECCV)*, Springer, 2012 214.

[209] P. F. Alcantarilla and T. Solutions, *Fast explicit diffusion for accelerated features in nonlinear scale spaces*, IEEE Transactions on Pattern Analysis and Machine Intelligence **34** (2011) 1281.

[210] D. Lang, D. W. Hogg, K. Mierle, M. Blanton and S. Roweis, *Astrometry. net: Blind astrometric calibration of arbitrary astronomical images*, The Astronomical Journal **139** (2010) 1782.

[211] M. A. Fischler and R. C. Bolles, *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*, Communications of the ACM **24** (1981) 381.

[212] Y. Ke and R. Sukthankar,
"PCA-SIFT: a more distinctive representation for local image descriptors", *Proceedings of the 2004 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2004 506.

[213] M. Doole, J. Ellerbroek and J. Hoekstra,
Estimation of traffic density from drone-based delivery in very low level urban airspace,
Journal of Air Transport Management **88** (2020) 101862.

[214] E. Lam, P. Le Bodic, D. D. Harabor and P. J. Stuckey,
"Branch-and-cut-and-price for multi-agent pathfinding",
Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI), 2019 1289.

[215] E. Lam, D. D. Harabor, P. J. Stuckey and J. Li, "Exact anytime multi-agent path finding using branch-and-cut-and-price and large neighborhood search", *Proceedings of the 33rd International Conference on Automated Planning and Scheduling (ICAPS)*, AAAI Press, 2023 254.

[216] R. Sadykov, F. Vanderbeck, A. Pessoa, I. Tahiri and E. Uchoa,
Primal heuristics for branch and price: The assets of diving methods,
INFORMS Journal on Computing **31** (2019) 251.

[217] Q. Li, Z. Huang, W. Jiang, Z. Tang and M. Song, *Quantum algorithms using infeasible solution constraints for collision-avoidance route planning*,
IEEE Transactions on Consumer Electronics (2024).

[218] M. Ali, H. Ahmed, M. H. Malik and A. Khalique,
Multicommodity information flow through quantum annealer,
Quantum Information Processing **23** (2024) 313.

[219] Y. Zhang, R. Zhang and A. C. Potter, *QED driven QAOA for network-flow optimization*,
Quantum **5** (2021) 510.

[220] S. Tarquini, D. Dragoni, M. Vandelli and F. Tudisco,
Testing quantum and simulated annealers on the drone delivery packing problem,
arXiv preprint arXiv:2406.08430 (2024).

[221] E. Davies and P. Kalidindi, "Quantum algorithms for drone mission planning",
Proceedings of Quantum Technologies for Defence and Security, SPIE, 2024 67.

[222] M. P. Martín and S. Martin,
"Unsplittable Multi-Commodity Flow Problem via Quantum Computing", *Proceedings of the 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, IEEE, 2023 385.

[223] Z. Huang, Q. Li, J. Zhao and M. Song,
Variational quantum algorithm applied to collision avoidance of unmanned aerial vehicles,
Entropy **24** (2022) 1685.

[224] E. Rönnberg and T. Larsson,
An integer optimality condition for column generation on zero–one linear programs,
Discrete Optimization **31** (2019) 79.

[225] D. Bertsekas, *Convex optimization algorithms*, Athena Scientific, 2015.

[226] G. B. Dantzig, *Linear programming*, Operations research **50** (2002) 42.

[227] J. Desrosiers and M. E. Lübbecke, *Branch-price-and-cut algorithms*, Encyclopedia of Operations Research and Management Science (2011) 109.

[228] S. Choudhury, K. Solovey, M. Kochenderfer and M. Pavone, "Coordinated Multi-Agent Pathfinding for Drones and Trucks over Road Networks", *Proceedings of the 21th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, IFAAMAS, 2022 272.

[229] J. Li et al., "Lifelong multi-agent path finding in large-scale warehouses", *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, AAAI Press, 2021 11272.

[230] G. Sharon, R. Stern, A. Felner and N. Sturtevant, "Conflict-Based Search For Optimal Multi-Agent Path Finding", *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, AAAI Press, 2012 563.

[231] J. Li, Z. Chen, D. Harabor, P. J. Stuckey and S. Koenig, "Anytime multi-agent path finding via large neighborhood search", *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*, 2021 4127.

[232] J. Li, Z. Chen, D. Harabor, P. J. Stuckey and S. Koenig, "MAPF-LNS2: Fast repairing for multi-agent path finding via large neighborhood search", *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI)*, AAAI Press, 2022 10256.

[233] T. Huang, J. Li, S. Koenig and B. Dilkina, "Anytime multi-agent path finding via machine learning-guided large neighborhood search", *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI)*, AAAI Press, 2022 9368.

[234] J. Y. Yen, *Finding the k shortest loopless paths in a network*, Management Science **17** (1971) 712.

[235] A. M. Geoffrion, "Lagrangean relaxation for integer programming", *Approaches to integer programming*, Springer, 2009 82.

[236] H. Ma, D. Harabor, P. J. Stuckey, J. Li and S. Koenig, "Searching with consistent prioritization for multi-agent path finding", *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*, AAAI Press, 2019 7643.

[237] M. F. Anjos and F. Liers, *Global approaches for facility layout and VLSI floorplanning*, Springer, 2012.

[238] D. Feld, *FieldPlacer-A flexible, fast and unconstrained force-directed placement method for heterogeneous reconfigurable logic architectures*, Fraunhofer Verlag, 2017.

[239] T. C. Koopmans and M. Beckmann,
Assignment problems and the location of economic activities,
Econometrica: Journal of the Econometric Society (1957) 53.

[240] V. Betz and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research",
International Workshop on Field Programmable Logic and Applications, Springer, 1997 213.

[241] A. Ludwin and V. Betz, *Efficient and deterministic parallel placement for FPGAs*,
ACM Transactions on Design Automation of Electronic Systems **16** (2011) 1.

[242] T.-H. Lin, P. Banerjee and Y.-W. Chang,
"An efficient and effective analytical placer for FPGAs",
Proceedings of the 50th Design Automation Conference (DAC), IEEE, 2013 1.

[243] M. A. Breuer, "A class of min-cut placement algorithms",
Proceedings of the 14th Design Automation Conference (DAC), IEEE, 1977 284.

[244] P. Maidee, C. Ababei and K. Bazargan,
Timing-driven partitioning-based placement for island style FPGAs, IEEE Transactions on
Computer-Aided Design of Integrated Circuits and Systems **24** (2005) 395.

[245] C.-W. Pui, G. Chen, Y. Ma, E. F. Young and B. Yu,
"Clock-aware ultrascale FPGA placement with machine learning routability prediction",
Proceedings of the 2017 International Conference on Computer-Aided Design (ICCAD),
IEEE, 2017 929.

[246] M. A. Elgamma, K. E. Murray and V. Betz,
"Learn to place: FPGA placement using reinforcement learning and directed moves",
Proceedings of the 2020 International Conference on Field-Programmable Technology (FPT),
IEEE, 2020 85.

[247] H. Bhatia et al., "CCuantuMM: Cycle-consistent quantum-hybrid matching of multiple shapes",
Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (ICCV), IEEE, 2023 1296.

[248] X. Guo, T. Wang, Z. Chen, L. Wang and W. Zhao,
"Fast FPGA placement algorithm using quantum genetic algorithm with simulated annealing",
Proceedings of the 8th International Conference on ASIC (ASICON), IEEE, 2009 730.

[249] A. U. Khalid, Z. Zilic and K. Radecka, "FPGA emulation of quantum circuits",
Proceedings of the 22nd IEEE International Conference on Computer Design (ICCD),
IEEE, 2004 310.

[250] J. Pilch and J. Długopolski, *An FPGA-based real quantum computer emulator*,
Journal of Computational Electronics **18** (2019) 329.

[251] Y. Xu et al., *QubiC: An open-source FPGA-based control and measurement system for superconducting quantum information processors*,
IEEE Transactions on Quantum Engineering **2** (2021) 1.

[252] H. Lütkepohl, *Handbook of matrices*, John Wiley & Sons, 1997.

[253] I. Rosenberg, *Reduction of bivalent maximization to the quadratic case*,
Cahiers du Centre d'Études de Recherche Opérationnelle **17** (1975) 71.

[254] E. Boros and A. Gruber, *On quadratization of pseudo-boolean functions*, arXiv preprint arXiv:1404.6538 (2014).

[255] A. Verma, M. Lewis and G. Kochenberger, *Efficient qubo transformation for higher degree pseudo boolean functions*, arXiv preprint arXiv:2107.11695 (2021).

[256] S. T. Rajavel and A. Akoglu, "An analytical energy model to accelerate FPGA logic architecture investigation", *Proceedings of the 2011 International Conference on Field-Programmable Technology (FPT)*, IEEE, 2011 1.

[257] S. Sahni and T. Gonzalez, *P-complete approximation problems*, Journal of the ACM (JACM) **23** (1976) 555.

[258] D. Shah et al., "Yosys+ nextpnr: an open source framework from verilog to bitstream for commercial fpgas", *Proceedings of the 27th International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, IEEE, 2019 1.

List of Figures

List of Tables